Palo Alto Research Center

Algorithm Schemata and Data Structures in Syntactic Processing

by Martin Kay

XEROX

Algorithm Schemata and Data Structures in Syntactic Processing

by Martin Kay

CSL-80-12 October 1980

© Xerox Corporation 1980

Abstract: The space in which models of human parsing strategy are to be sought is large. This paper is an exploration of that space, attempting to show what its dimensions are and what some of the choices are that the psycholinguist must face. Such an exploration as this may provide some protection against the common tendency to let some choices go by default.

A notion of *configuration tables* is used to locate algorithms on two dimensions according as (1) they work top-down or bottom-up, and (2) they are directed or undirected. The algorithms occupying a particular place in this two dimensional space constitute an *algorithm schema.* The notion of a *chart* is used to show how to limit the amount of work a parser must do by ensuring that nothing is done more than once. Finally, the notion of an *agenda* is introduced to show how a rich variety of psychological strategies can be combined in a principled way with a given algorithm schema to yield an algorithm.

A version of this paper will appear in the proceedings of the *Nobel Symposium on Text Processing*, Gothenburg, 1980.

CR Categories: 3.42, 3.36, 3.65, 5.23

Key words and phrases: Natural Language, Psycholinguistics, Parsing, Syntax.

XEROX
PALO ALTO RESEARCH CENTER
3333 Coyote Hill Road / Palo Alto / California 94304

1. INTRODUCTION

Interest in context-free grammars and associated parsing techniques has recently been rekindled among theoretical linguists and psycholinguists. A number of reasons for this suggest themselves. Chomsky's program for linguistics values a grammar more highly the more constrained the theoretical framework in which it is written. It is only by assuming that the human linguistic faculty is specialized for a narrow class of languages that we may hope to explain the prodigious feats of language learning accomplished by even the dullest of children in their first six years. So goes the argument. Generally, the most desirable constraints have been held to be those that place a theory low on the scale of weak generative power. In other words, the larger the class of languages that a theory cannot characterize at all—provided this class contains no natural languages—the better the theory.[1] However, determined attempts to modify tranformational grammar so as to reduce its weak generative power in a reasonably motivated manner have been largely unsuccessful. The overall theory remains relatively unconstrained. Bids for explanatory adequacy on its behalf continue to be based almost entirely on lists of *constraints*. These are more or less arbitrary from the point of view of the theory, and appended to it as riders and footnotes.

At the same time, it came to be recognized that the so called *proofs* made by early generative grammarians of the proposition that natural languages are not context free had been accepted far too eagerly (see, for example, Gazdar 1979a and 1979b). Context-free grammars occupy a low position in the hierarchy of weak generative power and they have the additional advantages of conceptual simplicity and susceptibility to algorithmic treatment. This last is not, of course, a requirement for a theory of linguistic competence but, *ceteris paribus*, a competence theory that also makes claims about the nature of linguistic performance is to be preferred.

Bresnan (1978) and others have pointed out that much of the machinery of transformations was not only unnecessary but also inappropriate even for such established and apparently natural uses as passivization. Bresnan proposed eliminating large numbers of transformational rules in favor of much weaker lexical redundancy rules. The only transformations that would remain according to this proposal would be *unbounded-movement* rules, such as topicalization and relativation, or possibly just a single *WH-movement* rule. With so much of the burden removed from it, the idea of eliminating the heavy machinery of transformations altogether becomes increasingly attractive.

At the same time, various new theories of grammar have been developed by computational linguists. While often not strictly context free, their grammars were of a sufficiently similar kind to be processed by minor variants of context-free algorithms. A sadly neglected early contribution to this line of development was that of Gilbert Harman (1963) whose theory made use of phrase-marker trees with complex symbols at the nodes and conventions by which features of dominating nodes were

[1]This is not an accurate characterization of Chomsky's own current view.

inherited by their descendents. Far more influential were the Augmented Transition Network (ATN) grammars of Woods and Kaplan (1971). These are not equivalent to context-free grammars but, like Harman's scheme, they arise by making relatively minor changes to the basic context-free formalism, mainly in the direction of permitting operations on complex symbols. Fairly straightforward and easily understood processing strategies can be used. Because of their susceptibility to algorithmic treatment, the ATN grammars have been proposed as a theory of linguistic performance (see, for example Kaplan (1972 and 1978), Wanner, Kaplan and Shiner (1975) and Wanner and Maratsos (1978).

A psychological model of sentence production or comprehension based on context-free grammar or a related formalism will seek to make predictions about the computational resources required for these activities. It must therefore be related to specific computational strategies. However, the number of parsing and generation algorithms available is large and there is no reason to suppose that it will not grow indefinitely. It seems, therefore, that a psycholinguist interested in syntax must conduct his investigations in a very large space having as three of its dimensions the linguistic theory, the particular language description within that theory, and the processing algorithms. In principle, at least, he may look to theoretical linguists for some guidance in exploring the first two of these dimensions. He should expect to receive some assistance with the third from computer scientists in general and computational linguists in particular. For the most part, however, very little such assistance has been forthcoming. I think there are two reasons for this.

First, most work by computer scientists, for example that quoted in Kimball (1973), has been done with a view to the formalization of programming languages and the automatic analysis of programs written in them. The first requirement of a programming language is that it should not admit syntactic, or indeed any other, ambiguities. Furthermore, the languages are usually designed so as to make it possible for a parser to make local choices on the basis of strictly limited contexts. As a result, the parsing algorithms that have been explored have been of a highly specialized kind, by no means appropriate for application to natural languages.

Second, computer scientists and computational linguists have usually approached questions of syntactic processing as engineers in search of algorithms that would achieve processing economies of one kind or another. While it is true that this line of attack could, in principle, produce plausible psychological models, there is no *a priori* reason to suppose that an investigation guided entirely by questions of efficiency on existing computers will ever lead in this direction. Griffiths and Petrick (1965) provided a useful classification of existing parsing algorithms and restatements of canonical members of each class in an ingenious form designed to highlight their salient features and to provide some basis for comparison, mostly from the point of view of computational efficiency.

If, as I shall argue at the end of this paper, the notion of an algorithm as normally conceived is an inappropriate component of a model of linguistic performance, then a great deal of groundwork must still be laid before any substantial results in the psycholinguistic investigation of syntax can

be expected. In some respects, my aim in this paper will be similar to that of Griffiths and Petrick (1965) in that I shall attempt to provide a conceptual vantage point from which to view a variety of different processing strategies and to assess the properties that distinguish them. I shall also discuss canonical representations of classes of algorithms. However, unlike Griffiths and Petrick, I shall not attempt to remove the mechanisms for achieving nondeterminism from the representations. In fact, I hold nondeterminism to be the crucial dimension along which systems that countenance ambiguity must be judged. Accordingly, there will be three major sections to the paper.

In the section 2, I shall review the principal classes of syntactic processing method that have been proposed[2] While it is true that my main concern will be with parsing, I shall begin with some consideration of sentence generation. I do this mainly to introduce some basic notions and to show that it is all of a piece with parsing when viewed from an appropriate vantage point. I shall concentrate heavily on those properties of a syntactic processor that guarantee that it will do everything that must be done when processing a sentence, leaving until later the question of how to avoid duplication of effort. Following a practice that has become common, especially in the psychological literature, I shall assume a fairly general back-up strategy to provide for non-determinism. To achieve a representation of classes of processing strategies, suppressing unnecessary detail, I shall make use of what I call *algorithm schemata* which differ from algorithms as usually conceived in that the sequence of events is not always uniquely specified, though the function eventually computed is.

Section 3 will be devoted to showing the error of so cavalier an approach to ambiguity and non-determinism. I shall show that it is not sufficient to simply assume some unspecified backup strategy and that, when this problem is given a central position in the assessment of processing techique, a different picture of the problem emerges. In particular, judgments of the probable computational costs of this or that strategy or grammatical device must be made afresh in the light of processing models that take nondeterminism seriously. The properties of a syntactic processor that enable it to avoid unnecessary computation steps will be a major concern in this section.

In section 4, I shall briefly consider how the algorithm schemata of the first two sections relate to specific algorithms and processing strategies. I shall urge the adoption of an algorithm schema, rather than any specific algorithm, in the construction of computer programs and psychological models of syntactic processing. The schema can be simulated by a particular algorithm, but this higher-order algorithm will have various properties that recommend it over the more well known first-order algorithms.

2. CONFIGURATION TABLES AND ALGORITHM SCHEMATA

For expository purposes, context-free grammars are usually described in terms of a *top-down*

[2]I shall not consider predictive analysis which, while it can be assimilated to the framework to be developed, requires grammars to be cast in a special form, thereby raising problems that are beyond the scope of this paper.

sentence-production process in which rules are applied to *derivation strings* to produce new derivation strings. The sequence of derivation strings involved is a *derivation* of the sentence. The process starts with a special derivation string consisting of just the initial symbol of the grammar. A derivation string to which no further rules can be applied is a *sentence.*

If it is required to obtain structures for the strings produced, one of a variety of modified versions of the basic procedure must be used. Suppose, for example, that syntactic structures, whether partial or complete, are represented by labeled bracketings, or *phrase markers*, $[\beta_1 \ldots \beta_k]_\alpha$ in which α is a non-terminal symbol and the β_i $(1 \leq i \leq k)$ are either terminal symbols or other labeled bracketings. Partial structures will contain subexpressions of the form $[?]_\alpha$, called *open boxes*, to indicate nonterminals whose substructure has yet to be developed. Given a symbol α, it will sometimes be convenient in the discussion that follows to use $\mathcal{F}(\alpha)$ to refer to the open box $[?]_\alpha$ in case α is non-terminal, and otherwise to α itself.

The production process starts with the expression $[?]_S$ on the first line of the derivation. A new line is derived from an old one by applying the *transition rule* (1):

(1) Select one open box $[?]_\alpha$ on the line and a rule $\alpha \rightarrow \beta_1 \ldots \beta_k$ from the grammar. Create a new configuration replacing the open box with $[\mathcal{F}(\beta_1) \ldots \mathcal{F}(\beta_k)]_\alpha$

Given the grammar shown in (2), such a *structural derivation* for the sentence *radio broadcasts pay* is shown in (3).

(2)

1. S → NP VP
2. NP → A N
3. NP → N
4. VP → V NP
5. VP → V
6. A → pay, radio, ...
7. N → broadcasts, pay, radio, ...
8. V → broadcasts, pay, radio, ...

Each line in (3) shows, in the *Rule* column, the number of the grammar rule by which it was obtained and, in the *Parent* column, the number of the line giving the immediately preceding state of the process. In this case, the entry in the *Parent* column always refers to the immediately preceding line. This will not be true of the tables we shall consider later.

(3)

#	Parent	Rule	Configuration
1	0		$[?]_S$
2	1	1	$[[?]_{NP}[?]_{VP}]_S$
3	2	2	$[[[?]_A[?]_N]_{NP}[?]_{VP}]_S$
4	3	6	$[[[\text{radio}]_A[?]_N]_{NP}[?]_{VP}]_S$
5	4	7	$[[[\text{radio}]_A[\text{broadcasts}]_N]_{NP}[?]_{VP}]_S$
6	5	5	$[[[\text{radio}]_A[\text{broadcasts}]_N]_{NP}[[?]_V]_{VP}]_S$
7	6	8	$[[[\text{radio}]_A[\text{broadcasts}]_N]_{NP}[[\text{pay}]_V]_{VP}]_S$

(4)

#	Par.	R	Configuration
1	0		$[?]_S$
2	1	1	$[[?]_{NP}[?]_{VP}]_S$
3	2	2	$[[[?]_A[?]_N]_{NP}[?]_{VP}]_S$
4	2	3	$[[[?]_N]_{NP}[?]_{VP}]_S$
5	3	6	$[[[pay]_A[?]_N]_{NP}{}^{VP?}]_S$
6	3	6	$[[[radio]_A[?]_N]_{NP}[?]_{VP}]_S$
7	4	7	$[[[broadcasts]_N]_{NP}[?]_{VP}]_S$
8	4	7	$[[[pay]_N]_{NP}[?]_{VP}]_S$
9	4	7	$[[[radio]_N]_{NP}[?]_{VP}]_S$
10	3	6	$[[[pay]_A[broadcasts]_N]_{NP}[?]_{VP}]_S$
11	3	6	$[[[pay]_A[pay]_N]_{NP}[?]_{VP}]_S$
12	3	6	$[[[pay]_A[radio]_N]_{NP}[?]_{VP}]_S$
13	7	4	$[[[broadcasts]_N]_{NP}[[?]_V[?]_{NP}]_{VP}]_S$
14	7	5	$[[[broadcasts]_N]_{NP}[[?]_V]_{VP}]_S$
15	8	4	$[[[pay]_N]_{NP}[[?]_V[?]_{NP}]_{VP}]_S$
16	8	5	$[[[pay]_N]_{NP}[[?]_V]_{VP}]_S$
17	9	4	$[[[radio]_N]_{NP}[[?]_V[?]_{NP}]_{VP}]_S$
18	9	5	$[[[radio]_N]_{NP}[[?]_V]_{VP}]_S$
19	10	4	$[[[pay]_A[broadcasts]_N]_{NP}[[?]_V[?]_{NP}]_{VP}]_S$
20	10	5	$[[[pay]_A[broadcasts]_N]_{NP}[[?]_V]_{VP}]_S$
21	11	4	$[[[pay]_A[pay]_N]_{NP}[[?]_V[?]_{NP}]_{VP}]_S$
22	11	5	$[[[pay]_A[pay]_N]_{NP}[[?]_V]_{VP}]_S$
23	12	4	$[[[pay]_A[radio]_N]_{NP}[[?]_V[?]_{NP}]_{VP}]_S$
24	12	5	$[[[pay]_A[radio]_N]_{NP}[[?]_V]_{VP}]_S$
25	13	8	$[[[broadcasts]_N]_{NP}[[broadcasts]_V[?]_{NP}]_{VP}]_S$
26	13	8	$[[[broadcasts]_N]_{NP}[[pay]_V[?]_{NP}]_{VP}]_S$
27	13	8	$[[[broadcasts]_N]_{NP}[[radio]_V[?]_{NP}]_{VP}]_S$
28	14	8	$[[[broadcasts]_N]_{NP}[[broadcasts]_V]_{VP}]_S$
29	14	8	$[[[broadcasts]_N]_{NP}[[pay]_V]_{VP}]_S$
30	14	8	$[[[broadcasts]_N]_{NP}[[radio]_V]_{VP}]_S$

The derivation process just described is not an algorithm in the strict sense of the word because it does not specifiy the sequence of steps to be followed exactly. Rather it is an *algorithm schema.* The transition rule (1) does not specify which open box should be chosen or which of the rules expanding α is to be applied. However, all algorithms derived from this schema by determining these choices generate the same sentences, though possibly with different derivations.

Suppose that the transition rule (1) is revised to read:

(5) Select one open box $[?]_\alpha$ on the line. For *each* rule $\alpha \rightarrow \beta_1 \ldots \beta_k$ in the grammar, create a new configuration, replacing the open box $[?]_\alpha$ with $[\mathcal{F}(\beta_1) \ldots \mathcal{F}(\beta_k)]_\alpha$

The result is an algorithm schema for producing all the sentences in the language of the grammar and the process will clearly terminate only in the degenerate case of a grammar like (2) that generates a finite language. The first thirty lines of a table generated according to this schema from the grammar (2) is shown in (4).

(3) and (4) are simple cases of *configuration tables.* Each line in such a table is a complete record

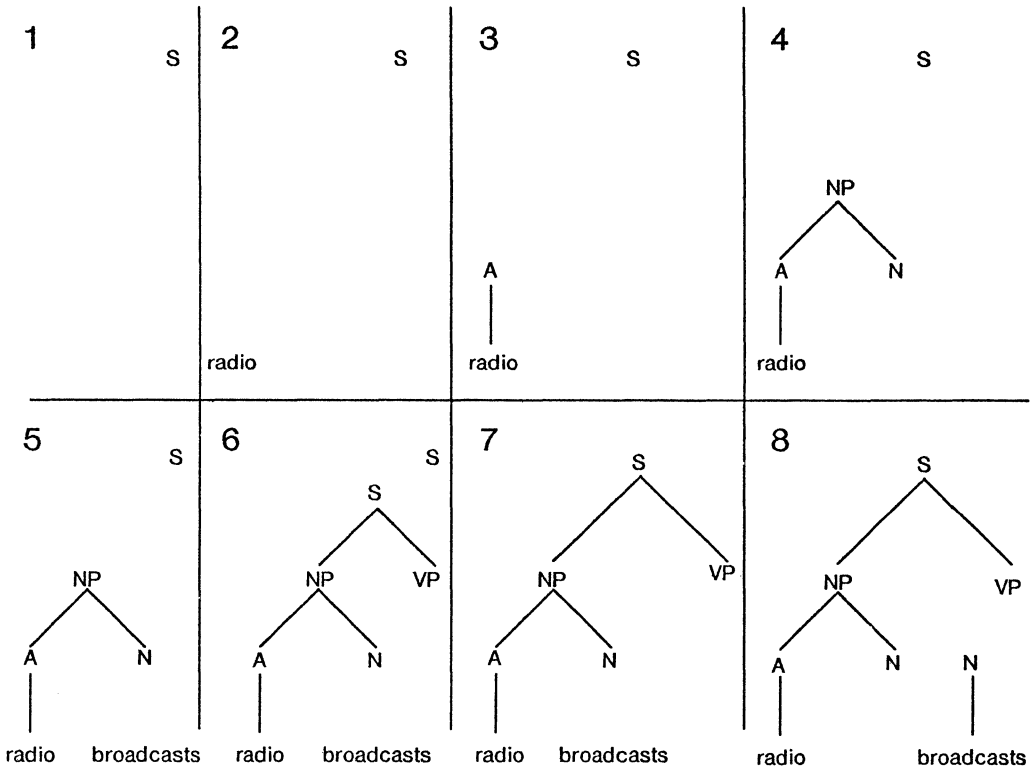
of the *state* of a process at a particular moment. In other words, each line either represents a final state in the process or contains all the information necessary to take the next step. The outputs of the computation chronicled in a configuration table are typically configurations that have a certain syntactic property. In the case of structural derivations like (3) and (4), a line containing no open boxes is part of the output. Generally speaking, several different steps can be taken independently from a given state, as in (4), and in these cases, the process as a whole is said to be *non-deterministic.* It is therefore no longer the case that each rule in a configuration table is derived from the one above it and for this reason a *Parent* column is printed in each table. It is there entirely for expository purposes, as is the *Rule* column, and plays no role in the algorithm. All the processes I shall examine below are non-deterministic.

It is also generally true that configuration tables belong to equivalence classes the members of each of which show the computation of a single function of the same inputs. There are two reasons for this. First, while all lines with a given parent must follow the parent line, the transition rules impose no constraint on the way they are ordered relative to one another. Second, the transition rules in accordance with which the process passes from state to state may be underdetermined, as in the case of the standard sentence-production derivations with which we started. (1) specifies that an open box is to be replaced by a new subexpression, but the choice of the open box to replace is open. There is an implicit claim that the results will be the same whichever choice is made.

Bottom-up Production

Sentence production can also be described in terms of *bottom-up* operations on *coverings.* A covering is simply a sequence of phrases made up of the words in the string. So, for example, the sentence *I know the girl he married* has one covering consisting of a pronoun, a verb, and a noun phrase, and another consisting of a pronoun, a verb, two noun phrases, and another verb. It also has others. More precisely, a covering of a string is a sequence of phrase markers the labels on whose terminal nodes, when concatenated in the order of the sequence, constitute the string. It will usually be implicit that the coverings we are considering are constructed according to the rules of a particular grammar. The string of symbols obtained by reading off the root labels from a covering is a *root string* so that the two coverings of the sentence *I know the girl he married* just mentioned correspond to the root strings *PRON V NP* and *PRON V NP NP V.* Clearly every derivation string relative to a grammar is the root string for some covering, but covering strings are not necessarily also derivation strings. For example, *I know the girl he married* can be seen as the concatenation of the sentence *I know* and the topicalized sentence *the girl he married.* Accordingly, it has a covering whose root string is *S S* even though this would presumably not be a line in its derivation by any reasonable grammar. Similarly, relative to the grammar (2), the sentence *radio broadcasts pay*

(6)



has a covering consisting of a sentence ($[[[\text{radio}]_N]_{NP}[[\text{broadcasts}]_V]_{VP}]_S$) followed by verb phrase ($[[\text{pay}]_V]_{VP}$), but no derivation *S VP*.

If the root string of a covering relative to a given grammar contains a substring that matches the right-hand side of a rule, a new covering relative to that grammar can be produced by replacing the corresponding subsequence of trees with a single tree whose root is labeled with the symbol on the left-hand side of the rule and whose immediate descendants are that subsequence of trees. The new covering is a *reduction* of the original one. Consider the sentence *I know the girl he married* once more. As we saw, an intuitively reasonable grammar would give it a covering with the root string *PRON V NP NP V*. Assuming a rule that rewrites *VP*'s as *V*'s, *PRON V NP NP V*,*P* and *PRON VP NP NP V* are both reductions of this. Assuming another rule that expands *S* into *NP NP V*, the first of these has the further reduction *PRON V S*. The bottom-up procedure aims to produce a string from which a root string consisting of the initial symbol of the grammar can be obtained by a series of reductions.

A fairly straightforward method of bottom-up generation makes use of a *reachability* table which shows, for every non-terminal symbol α in a grammar, the set $\mathcal{R}(\alpha)$ of terminal and non-terminal symbols each of which can be the first element in a string dominated by α. The reachability table

for the grammar in (2) is as follows:

(7)

α	$\mathcal{R}(\alpha)$
S	NP, A, N, broadcasts, pay, radio
NP	A, N, broadcasts, pay, radio
VP	V, broadcasts, pay, radio
A	pay, radio, ...
N	broadcasts, pay, radio, ...
V	broadcasts, pay, radio, ...

The overall goal is to construct a sentence and accordingly the initial configuration consists of the open box $[?]_S$. Whereas in (3) and (4), a configuration is a single phrase marker, we shall henceforth be working with configurations that are sequences of *terms* each of which is a phrase marker. A term that contains at least one open box will be called *active*; others are *inactive*. The interpretation is as follows: the part of the sequence to the right of a given term shows the current state of progress towards filling the first open box in that term. So, for example, in the sequence $[?]_S[?]_{NP}[\text{radio}]_N$, the noun *radio* will fill the open box in the noun phrase, either alone or with other material yet to be obtained, and that will go towards filling the sentence. Thus, a well formed configuration can contain at most one inactive term which must be in the final position.

The sequence of terms that make up a configuration are portions of a single tree structure that will eventually be constructed. The top node first term in a configuration will become the root of the final tree. Nonterminal symbols in a term that have no structure below them are the open boxes. Terms that contain open boxes are "looking for" other terms to dominate and they are therefore said to be *active*. If one term precedes another in a configuration, the latter will eventually come to be dominated by one such nonterminal in the former, that is, it will "fill the open box". (6) gives a graphic representation of the first eight configurations in the table (9). Notice that lines 5 and 6 in (9) each contain three terms and that squares 5 and 6 in (6) each contain three disconnected subtrees. Left-to-right ordering in the table corresponds to top-to-bottom ordering in the diagrams. The aim of the procedure will always be to conflate pairs of adjacent subtrees, building new structure below the upper one or above the lower one when necessary.

Suppose $[?]_\alpha$ is the first open box of the last term in the configuration. Using the reachability table, a terminal t is chosen that could be initial in a phrase of type α. If the grammar contains the rule $\alpha \rightarrow t$, then t can simply be used to fill the open box. In other words, a new configuration is created replacing the last two terms by $[t]_\alpha$ (See, for example, lines 3, 8, and 11 in (9)). Otherwise, a rule of the form $\beta \rightarrow \gamma_1 \ldots \gamma_k$ is found in the grammar, such that β can be initial in a phrase of type α and $\gamma_1 = t$. t is replaced as the last term in the configuration by $[\mathcal{F}(\gamma_1)\ldots\mathcal{F}(\gamma_k)]_\beta$. A rule of the required form must exist, given the way the terminal was chosen using the reachability table. If the term contains other open boxes, new terminals are chosen and the process repeats until they are all full. If the last term in a configuration contains no open boxes, it is inactive and is treated as

though it were the most recently chosen terminal. The aim now becomes that of incorporating this term in the one that immediately precedes it in the configuration.

In summary, the process is governed by three transition rules:

(8) 1. **Append a terminal symbol:** If the last term in the configuration is active, let $[?]_\alpha$ be the first open box it contains. Select a terminal $t \in \mathcal{R}(\alpha)$ and create a new configuration by appending t to the current one.

2. **Find a dominating non-terminal:** If the last term is inactive, consider two adjacent terms g_i and g_{i+1}, of the current configuration. If g_{i+1} is of category α (i.e. it consists of the terminal α or an expression of the form $[\ldots]_\alpha$) and $[?]_\beta$ is the first open box in g_i, for each rule $\gamma \rightarrow \delta_1 \ldots \delta_n$ such that $\gamma = \beta$ or $\gamma \in \mathcal{R}(\beta)$ and $\alpha = \delta_1$, create a new configuration replacing g_{i+1} with $[g_{i+1}\ \mathcal{F}(\delta_2) \ldots \mathcal{F}(\delta_n)]_\gamma$.

3. **Incorporate completed subgoal:** If the last term is inactive, consider two adjacent terms g_i and g_{i+1}, of the current configuration. If g_{i+1} is of category α and the first open box in g_i is $[?]_\alpha$, replace that open box with g_{i+1} and delete g_{i+1} from the configuration.

Rule 1 can apply only in circumstances in which rules 2 and 3 do not. Rules 2 and 3, however, do not exclude one another and, when both apply to the same configuration, the values chosen for i must be the same. The reason is easy to see. The application of each destroys the conditions of applicability of the other so that, if both lines of attack are to be explored, the rules must be applied simultaneously. In the examples in this paper, i is systematically chosen to be as large as possible. One result of this, which will be of interest in the next section, is that rules 2 and 3 will invariably be applied to a pair of terms the first of which is active, as the rules require, and the second of which is inactive. But if rules 2 or 3 can be applied in more than one place, that is, for more than one value of i, then only one is chosen and the choice is arbitrary. This is because applications of the rules to different terms is independent. To see that this is true, consider three adjacent terms $\ldots t_i\ t_{i+1}\ t_{i+2} \ldots$ of a configuration and suppose that transition rule 2 applies to both pairs $< t_i, t_{i+1} >$ and $< t_{i+1}, t_{i+2} >$. If it is applied to the first pair, t_{i+1} will be replaced by a new term u of the form $[t_{i+1}]_\alpha$, for some α. Since t_{i+1} is the first term dominated by α in u, the first open box in u is the same as the first open box in t_{i+1}. Applying rule 2 to the pair $< u, t_{i+2} >$ will therefore be the same as applying it to $< t_{i+1}, t_{i+2} >$. Suppose, on the other hand, that rule 2 is applied first to $< t_{i+1}, t_{i+2} >$. In this case, t_{i+1} is unchanged and application to the first pair is therefore unaffected. An exactly parallel argument applies to rule 3 and to combinations of the two.

(9) is a configuration table showing how these transition rules can be used to generate the sentence *radio broadcasts pay*, in one of its syntactic readings, using the grammar (2). The column headed *T. R.* gives the number of the transition rule employed.

(9)

#	T. R.	Par.	R	Configuration
1				$[?]_S$
2	1	1		$[?]_S$radio
3	2	2	6	$[?]_S[\text{radio}]_A$
4	2	3	2	$[?]_S[[\text{radio}]_A[?]_N]_{NP}$
5	1	4		$[?]_S[[\text{radio}]_A[?]_N]_{NP}$broadcasts
6	2	5	1	$[?]_S[[[\text{radio}]_A[?]_N]_{NP}[?]_{VP}]_S$broadcasts
7	3	6		$[[[\text{radio}]_A[?]_N]_{NP}[?]_{VP}]_S$broadcasts
8	2	7	7	$[[[\text{radio}]_A[?]_N]_{NP}[?]_{VP}]_S[\text{broadcasts}]_N$
9	3	8		$[[[\text{radio}]_A[\text{broadcasts}]_N]_{NP}[?]_{VP}]_S$
10	1	9		$[[[\text{radio}]_A[\text{broadcasts}]_N]_{NP}[?]_{VP}]_S$pay
11	2	10	8	$[[[\text{radio}]_A[\text{broadcasts}]_N]_{NP}[?]_{VP}]_S[\text{pay}]_V$
12	2	11	5	$[[[\text{radio}]_A[\text{broadcasts}]_N]_{NP}[?]_{VP}]_S[[\text{pay}]_V]_{VP}$
13	3	12		$[[[\text{radio}]_A[\text{broadcasts}]_N]_{NP}[[\text{pay}]_V]_{VP}]_S$

Bottom-up Parsing

Bottom-up sentence production is something of a *tour de force.* It is essentially a procedure of *synthesis by analysis*—a parsing procedure in which the string is constructed so as to ensure that the first hypothesis made about its structure at each step turns out right. The fully fledged parsing procedure can be derived from that just given with very few modifications. This time, however, the process is non-deterministic because it must countenance ambiguities.

I shall follow the usual practice in discussing parsing procedures of assuming that the terminals have been looked up in the lexicon in a previous step so that the data consists not of a string of terminals but a string of alternative lexical entries each of which is a family of degenerate phrase markers of the form $[t]_\alpha$, where t is a terminal and α a *pre-terminal* symbol. After lexical lookup, the sentence *radio broadcasts pay* takes on essentially the following form.

$$\left\{\begin{matrix}[\text{radio}]_A\\ [\text{radio}]_N\\ [\text{radio}]_V\end{matrix}\right\} \quad \left\{< \begin{matrix}[\text{broadcasts}]_V\\ [\text{broadcasts}]_N\end{matrix}\right\} \quad \left\{\begin{matrix}[\text{pay}]_A\\ [\text{pay}]_N\\ [\text{pay}]_V\end{matrix}\right\}$$

Only the first two transition rules of the production procedure in (8) require modification. They must now read:

(10) 1. **Append a terminal symbol:** If the last goal in the list is active, let $[?]_\alpha$ be the first open box that it contains. For each lexical entry $[w_i]_\beta$ of the next word in the string, w_i, such that $\beta = \alpha$ or $\beta \in \mathcal{R}(\alpha)$, create a new line in the table in which the open box is replaced by $[w_i]_\beta$.

2. **Find a dominating non-terminal:** Consider two adjacent terms g_i and g_{i+1}, of the current configuration. If g_{i+1} is of category α and $[?]_\beta$ is the first open box in g_i, then for all rules $\gamma \rightarrow \delta_1 \ldots \delta_n$ such that $\gamma = \beta$ or $\gamma \in \mathcal{R}(\beta)$, and $\alpha = \delta_1$, replace g_{i+1} with $[g_{i+1}\ \mathcal{F}(\delta_2) \ldots \mathcal{F}(\delta_n)]_\gamma$.

(11)

#	T.R.	Pos.	Par.	R	Configuration
1		0	0		$[?]_S$
2	1	1	1		$[?]_S[\text{radio}]_A$
3	1	1	1		$[?]_S[\text{radio}]_N$
4	2	1	2	2	$[?]_S[[\text{radio}]_A[?]_N]_{NP}$
5	2	1	3	3	$[?]_S[[\text{radio}]_N]_{NP}$
6	1	2	4		$[?]_S[[\text{radio}]_A[?]_N]_{NP}[\text{broadcasts}]_N$
7	2	1	5	1	$[?]_S[[[\text{radio}]_N]_{NP}[?]_{VP}]_S$
8	3	2	6		$[?]_S[[\text{radio}]_A[\text{broadcasts}]_N]_{NP}$
9	1	2	7		$[?]_S[[[\text{radio}]_N]_{NP}[?]_{VP}]_S[\text{broadcasts}]_V$
10	2	2	8	1	$[?]_S[[[\text{radio}]_A[\text{broadcasts}]_N]_{NP}[?]_{VP}]_S$
11	2	2	9	4	$[?]_S[[[\text{radio}]_N]_{NP}[?]_{VP}]_S[[\text{broadcasts}]_V[?]_{NP}]_{VP}$
12	2	2	9	5	$[?]_S[[[\text{radio}]_N]_{NP}[?]_{VP}]_S[[\text{broadcasts}]_V]_{VP}$
13	1	3	10		$[?]_S[[[\text{radio}]_A[\text{broadcasts}]_N]_{NP}[?]_{VP}]_S[\text{pay}]_V$
14	1	3	11		$[?]_S[[[\text{radio}]_N]_{NP}[?]_{VP}]_S[[\text{broadcasts}]_V[?]_{NP}]_{VP}[\text{pay}]_A$
15	1	3	11		$[?]_S[[[\text{radio}]_N]_{NP}[?]_{VP}]_S[[\text{broadcasts}]_V[?]_{NP}]_{VP}[\text{pay}]_N$
16	3	2	12		$[?]_S[[[\text{radio}]_N]_{NP}[[\text{broadcasts}]_V]_{VP}]_S$
17	2	3	13	4	$[?]_S[[[\text{radio}]_A[\text{broadcasts}]_N]_{NP}[?]_{VP}]_S[[\text{pay}]_V[?]_{NP}]_{VP}$
18	2	3	13	5	$[?]_S[[[\text{radio}]_A[\text{broadcasts}]_N]_{NP}[?]_{VP}]_S[[\text{pay}]_V]_{VP}$
19	2	3	14	2	$[?]_S[[[\text{radio}]_N]_{NP}[?]_{VP}]_S[[\text{broadcasts}]_V[?]_{NP}]_{VP}[[\text{pay}]_A[?]_N]_{NP}$
20	2	3	15	3	$[?]_S[[[\text{radio}]_N]_{NP}[?]_{VP}]_S[[\text{broadcasts}]_V[?]_{NP}]_{VP}[[\text{pay}]_N]_{NP}$
21	3	2	16		$[[[\text{radio}]_N]_{NP}[[\text{broadcasts}]_V]_{VP}]_S$
22	3	3	18		$[?]_S[[[\text{radio}]_A[\text{broadcasts}]_N]_{NP}[[\text{pay}]_V]_{VP}]_S$
23	3	3	20		$[?]_S[[[\text{radio}]_N]_{NP}[?]_{VP}]_S[[\text{broadcasts}]_V[[\text{pay}]_N]_{NP}]_{VP}$
24	3	3	22		$[[[\text{radio}]_A[\text{broadcasts}]_N]_{NP}[[\text{pay}]_V]_{VP}]_S$
25	3	3	23		$[?]_S[[[\text{radio}]_N]_{NP}[[\text{broadcasts}]_V[[\text{pay}]_N]_{NP}]_{VP}]_S$
26	3	3	25		$[[[\text{radio}]_N]_{NP}[[\text{broadcasts}]_V[[\text{pay}]_N]_{NP}]_{VP}]_S$

Rule 1 adds a terminal—or more strictly, a preterminal—to the end of a configuration whose current last term contains open boxes, that is, that represents a subtree with nonterminal symbols that dominate no substructure. The only lexical interpretations of the terminal that need be considered are those that can be dominated by the first nonterminal in the subtree that is still in need of substructure. Rule 2 builds some additional structure on top of the lower of a pair of adjacent subtrees with a view to eventually conflating them (by rule 3).

Complete analyses appear in the table as lines that contain no active terms. (11) is a configuration table showing the analysis of the sentence *radio broadcasts pay* using the grammar in (2). The entries in the Pos. column show the serial number of the last terminal considered. Where transition rule 1 has been applied, the number of the relevant grammar rule is also given.

In transition rule 2, rules of grammar are chosen in accordance with two criteria: (i) The symbol on the left-hand side of the rule must be reachable from the symbol beneath which it is eventually to be inserted ($\gamma = \beta$ or $\gamma \in \mathcal{R}(\beta)$) and (ii) the first symbol on the right-hand side of the rule must match a particular symbol already established in the configration ($\alpha = \delta_1$). It is the second of these criteria that is characterisitic of a bottom-up process since it ensures that new non-terminals are introduced only when at least one branch connecting them to a terminal symbol has already been constructed. The first criterion characterizes the process as *directed* and it serves only to prevent the

creation of certain phrases for which there is known to be no place in a higher-level structure. This process is also directed to the extent that transition rule 1 requires the terminal selected to be in the reachability set of a particular non-terminal symbol ($\beta = \alpha$ or $\beta \in \mathcal{R}(\alpha)$). The *undirected* process that results from removing these conditions, though less efficient, is not only viable, but represents the type of bottom-up procedure that has most often been proposed. The complete set of transition rules for an undirected bottom-up parser would be as in (12):

(12) 1. **Append a terminal symbol:** If the last term in the configuration is active and there remain terminals in the string to be considered, create a new configuration for each lexical alternative of the next one and append it to the end of the configuration.

2. **Find a dominating non-terminal:** If the last term is inactive and g_i is of category α and a term of the current configuration, other than the last, then for all rules $\gamma \rightarrow \delta_1 \ldots \delta_n$ such that $\alpha = \delta_1$, replace g_i with $[g_i\ \mathcal{F}(\delta_2) \ldots \mathcal{F}(\delta_n)]_\gamma$. This operation is inhibited if the sum of terminals and open boxes in a new configuration would be greater than the number of words in the string being analyzed.

3. **Incorporate completed subgoal:** If the last term is inactive, consider two adjacent terms g_i and g_{i+1}, of the current configuration. If g_{i+1} is of category α and the first open box in g_i is $[?]_\alpha$, replace that open box with g_{i+1} and delete g_{i+1} from the list.

The modification to rule 2 that inhibits the production of configurations with more open boxes than can be filled by the remainder of the string under analysis is called the *shaper* test and is due to Kuno and Oettinger (1964). It is needed here to ensure that the process will terminate even if the grammar contains *left-recursive* groups of rules. A rule of the form $\alpha \rightarrow \alpha \ldots$ constitutes such a group by itself. In general, a left-recursive group is of the form

$$\begin{aligned}
\alpha_1 &\rightarrow \alpha_2 \ldots \\
\alpha_2 &\rightarrow \alpha_3 \ldots \\
&\cdot \quad \rightarrow \quad \cdot \\
&\cdot \quad \rightarrow \quad \cdot \\
&\cdot \quad \rightarrow \quad \cdot \\
\alpha_i &\rightarrow \alpha_1 \ldots
\end{aligned}$$

Top-down strategies are notoriously unstable in the presence of such rules unless restrained by something like the shaper test. We make the usual assumption that the grammar contains no cyclic sets of nonbranching rules, that is, there are no sets of the form:

$$\begin{array}{lll} \alpha_1 & \rightarrow & \alpha_2 \\ \alpha_2 & \rightarrow & \alpha_3 \\ . & \rightarrow & . \\ . & \rightarrow & . \\ . & \rightarrow & . \\ \alpha_i & \rightarrow & \alpha_1 \end{array} \qquad s$$

The initial configuration consisting of the single term $[?]_S$ can be usefully eliminated from the configuration table for undirected bottom-up parsing provided that a configuration is entered in the table for each lexical entry of the first item in the string before the process begins. The complete configuration table for *radio broadcasts pay* using grammar (2) contains 72 configurations. A more modest example appears later in the paper at (25).

Top-down Parsing

We come finally to top-down parsing which is illustrated, in its directed variety, in (15). As before, the initial configuration is $[?]_S$. The transition rules are the same as in (10) except for the second which now reads as follows:

(13) 2. **Find a dominating non-terminal:** If the last term is inactive, consider two adjacent terms g_i and g_{i+1}, of the current configuration. If g_{i+1} is of category α and $[?]_\beta$ is the first open box in g_i, then for all rules $\beta \rightarrow \gamma_1 \ldots \gamma_n$ such that $\alpha = \gamma_1$ or $\alpha \in \mathcal{R}(\gamma_1)$, replace the first open box $[?]_\beta$ in g_i with $[g_{i+1}\ \mathcal{F}(\gamma_2) \ldots \mathcal{F}(\gamma_n)]_\beta$. As before, the operation is inhibited if the sum of terminals and open boxes in a new configuration would be greater than the number of words in the string being analyzed.

An undirected top-down parser differs in that the selection of the grammar rule to be applied does not make use of the reachability table. The appropriate transition rule is:

(14) 2. **Find a dominating non-terminal:** If the last term is inactive and g_i is a term in the current configuration and $[?]_\beta$ is the first open box that it contains, then for all rules $\beta \rightarrow \gamma_1 \ldots \gamma_n$, replace the open box $[?]_\beta$ with $[\mathcal{F}(\gamma_2) \ldots \mathcal{F}(\gamma_n)]_\beta$. The operation is inhibited if the sum of terminals and open boxes in a new configuration would be greater than the number of words in the string being analyzed.

Rule-Selection Tables

When expressed in terms of configuration tables, the differences between bottom-up and top-down parsing appear quite small. In (11), the transition from line 2 to line 4 uses rule 2 because the terminal at the end of the configuration on line 2 is of category A and rule 2 permits this to be

(15)

#	T.R.	Pos.	Par.	R	Configuration
1		0	0		$[?]_S$
2	1	1	1		$[?]_S[\text{radio}]_A$
3	1	1	1		$[?]_S[\text{radio}]_N$
4	2	1	2	1	$[[?]_{NP}[?]_{VP}]_S[\text{radio}]_A$
5	2	1	3	1	$[[?]_{NP}[?]_{VP}]_S[\text{radio}]_N$
6	2	1	4	2	$[[[?]_A[?]_N]_{NP}[?]_{VP}]_S[\text{radio}]_A$
7	2	1	5	3	$[[[?]_N]_{NP}[?]_{VP}]_S[\text{radio}]_N$
8	3	1	6		$[[[\text{radio}]_A[?]_N]_{NP}[?]_{VP}]_S$
9	3	1	7		$[[[\text{radio}]_N]_{NP}[?]_{VP}]_S$
10	1	2	8		$[[[\text{radio}]_A[?]_N]_{NP}[?]_{VP}]_S[\text{broadcasts}]_N$
11	1	2	9		$[[[\text{radio}]_N]_{NP}[?]_{VP}]_S[\text{broadcasts}]_V$
12	3	2	10		$[[[\text{radio}]_A[\text{broadcasts}]_N]_{NP}[?]_{VP}]_S$
13	2	2	11	4	$[[[\text{radio}]_N]_{NP}[[?]_V[?]_{NP}]_{VP}]_S[\text{broadcasts}]_V$
14	2	2	11	5	$[[[\text{radio}]_N]_{NP}[[?]_V]_{VP}]_S[\text{broadcasts}]_V$
15	1	3	12		$[[[\text{radio}]_A[\text{broadcasts}]_N]_{NP}[?]_{VP}]_S[\text{pay}]_V$
16	3	2	13		$[[[\text{radio}]_N]_{NP}[[\text{broadcasts}]_V[?]_{NP}]_{VP}]_S$
17	3	2	14		$[[[\text{radio}]_N]_{NP}[[\text{broadcasts}]_V]_{VP}]_S$
18	2	3	15	5	$[[[\text{radio}]_A[\text{broadcasts}]_N]_{NP}[[?]_V]_{VP}]_S[\text{pay}]_V$
19	1	3	16		$[[[\text{radio}]_N]_{NP}[[\text{broadcasts}]_V[?]_{NP}]_{VP}]_S[\text{pay}]_A$
20	1	3	16		$[[[\text{radio}]_N]_{NP}[[\text{broadcasts}]_V[?]_{NP}]_{VP}]_S[\text{pay}]_N$
21	3	3	18		$[[[\text{radio}]_A[\text{broadcasts}]_N]_{NP}[[\text{pay}]_V]_{VP}]_S$
22	2	3	20	3	$[[[\text{radio}]_N]_{NP}[[\text{broadcasts}]_V[[?]_N]_{NP}]_{VP}]_S[\text{pay}]_N$
23	3	3	22		$[[[\text{radio}]_N]_{NP}[[\text{broadcasts}]_V[[\text{pay}]_N]_{NP}]_{VP}]_S$

immediately incorporated in a higher-level phrase, namely an NP. The reachability table guarantees that there is a place for this at the beginning of a sentence. A top-down strategy applies a rule that expands the symbol S, secure in the knowledge that the first symbol on the right-hand side of that rule can have an adjective on its left-most branch. Notice, however, that the only difference between the two strategies resides in the way the rules are chosen and not in any other detail of how the configuration table develops. A single set of transition rules could be written for both kinds of procedure if it were made to depend on a *Rule-Selection Table* instead of a reachability table.

A Rule-Selection Table is an $m \times n$ array, where m is the number of non-terminal symbols in the grammar that are not pre-terminals and n is the total number of non-terminals. The cell $S_{\alpha,\beta}$ of a rule-selection table S contains the list of grammar rules to be used in transition rule 2 when α is the *upper* symbol and β is the *lower*. In other words, α is the label of the first open box in a term and the immediately following term in the configuration is $\mathcal{F}(\beta)$. For a top-down procedure, these rules will all be of the form $\alpha \rightarrow \gamma_1 \ldots \gamma_n$. If the procedure is undirected, all rules whose left-hand symbol is α will be on the list. If it is directed, the list will contain only those for which $\beta = \gamma_1$ or $\beta \in \mathcal{R}(\gamma_1)$. (16) is a rule-selection table specifying a directed top-down procedure for grammar (2). The directed top-down procedure would have 2,3 in place of 2 and 3 in the NP row. The table for an undirected top-down procedure has the same value in every non-null cell of a given row. Conversely, the table for an undirected bottom-up procedure procedure has the same value in every non-null cell of a given column. For a bottom-up procedure, the rules will all be of the form

$\gamma \rightarrow \delta_1 \ldots \delta_n$ where $\delta_1 = \beta$. In the directed case, only rules for which $\gamma = \alpha$ or $\gamma \in \mathcal{R}(\alpha_1)$ will be included.

(16)

	S	NP	VP	A	N	V
S		1		1	1	
NP				2	3	
VP						4,5
A						
N						
V						

We can now write a general set of transition rules that implements whatever strategy the rule-selection table provided to it dictates:

(17) 1. **Append a terminal symbol:** If the last term in the configuration is active, let $[?]_\alpha$ be the first of them. If there remain terminals in the string to be considered, then for each lexical alternative of the next terminal $[t]_\beta$ such that $\beta \in \mathcal{R}(\alpha)$ that can be on a left branch of α, create a new configuration by appending $[t]_\beta$ to the end of the current one.

2. **Find a dominating non-terminal:** If the last term is inactive, consider two adjacent terms g_i and g_{i+1}, of the current configuration. If $g_{i+1} = \mathcal{F}(\alpha)$ and $[?]_\beta$ is the first open box in g_i, for each rule $\gamma \rightarrow \delta_1 \ldots \delta_n$ in $\mathcal{S}_{\beta,\alpha}$, create a new configuration introducing $[\mathcal{F}(\delta_1) \ldots \mathcal{F}(\delta_n)]_\gamma$ between g_i and g_{i+1}.

3. **Incorporate completed subgoal:** If the last term is inactive, consider two adjacent terms g_i and g_{i+1}, of the current configuration. If $g_{i+1} = \mathcal{F}(\alpha)$ and the first open box in g_i is $[?]_\alpha$, replace that open box with g_{i+1} and delete g_{i+1} from the list.

A word of explanation is in order about the second of these transition rules which, unlike corresponding transition rules in previous schemes, is formulated so as to introduce a new term that does not replace any existing ones. In the top-down scheme, it was possible to eliminate an open box in this step because the rules were chosen precisely to provide contents for such a box. In the bottom-up scheme, an existing item could be incorporated into the newly introduced structure because the structures were chosen with a view to making that possible. However, the rules supplied by the rule-selection table may be such as to allow neither possibility. We therefore leave the incorporation of existing material to a later application of rule 3. An alternative solution would have been to recognize three cases of rule 2 according as (i) $\beta = \gamma$, (ii) $\alpha = \delta_1$ but $\beta \neq \gamma$, or (ii) neither of the above. Clearly, it would also be possible to give special treatment to the case in which $\beta = \gamma$ and $\alpha = \delta_1$ for, under these circumstances both g_i and g_{i+1} could be incorporated in the new structure. The solution I have adopted, while it makes for more lines in the configuration table, is the simplest. More important, we shall see that it leads to important generalizations in the next section.

(18)

	S	NP	VP	A	N	V
S		1		2	1	
NP				2	3	
VP						4,5
A						
N						
V						

By now it should be clear that the distinction between top-down and bottom-up procedures is not primary. Rule tables can generally be assembled in a large number of ways only some of which correspond to either top-down or bottom-up procedures. Perhaps these procedures should be characterized as *middle-out*. (18) is a trivial example, based on (2). According to this table, sentences beginning with N will be analyzed by a procedure whose first move is top-down, whereas the analysis of those beginning with A will begin bottom-up.

The conditions under which a rule-selection table properly represents a given grammar are quite straightforward. Clearly, the rules in the cell $S_{\alpha,\beta}$ must be chosen from the set $\mathcal{G}_{\alpha,\beta} = \{\gamma \rightarrow \delta_1 \ldots \delta_n \mid \gamma \in \mathcal{R}(\alpha) \,.\, \beta \in \mathcal{R}(\delta_1)\}$. In other words, the symbol on the left-hand side of each rule must be reachable from the upper symbol and the lower symbol must be reachable from the first symbol on the right-hand side of the rule. Suppose now that $\mathcal{G}_{\alpha,\beta}$ contains the pair

(19)
$$\kappa \rightarrow \lambda_1 \ldots \lambda_m$$
$$\mu \rightarrow \nu_1 \ldots \nu_n$$

and that

(20)
$$\mu \in \mathcal{R}(\lambda_1)$$

Only one member of the pair must be included in $\mathcal{R}_{\alpha,\beta}$ because leaving both in the set will give rise to redundancies in the analysis process. To see that this is true, it is sufficient to consider how they would apply to a configuration of the form

$$\ldots [?]_\alpha \, \mathcal{F}(\beta) \ldots$$

Clearly two new configurations would be produced as follows:

(21)
$$\ldots [?]_\alpha \, [[?]_{\lambda_1} \, \mathcal{F}(\lambda_2) \ldots \mathcal{F}(\lambda_m)]_\kappa \, \mathcal{F}(\beta) \ldots$$

(22)
$$\ldots [?]_\alpha \, []_\mu \mathcal{F}(\nu_1) \ldots \mathcal{F}(\nu_n) \, \mathcal{F}(\beta) \ldots$$

but, since $\mu \in \mathcal{R}(\lambda_1)$, (22) must also arise, directly or indirectly from (21).

Proper values for $S_{\alpha,\beta}$ are therefore those that are derived from $\mathcal{G}_{\alpha,\beta}$ by deleting one member of any pair of rules standing in the relation (19) and (20) and continuing to do this until no such pairs

remain. The table corresponding to a bottom-up strategy is arrived at by systematically removing the rule corresponding to the first member of the pair in (19) and the top-down strategy, by removing the second member.

The Efficiency Question

It is often supposed that the most efficient parsers use top-down methods. So long as the discussion is confined to undirected strategies, there is doubtless some broad statistical validity to the claim though it is easy to construct grammars and strings that falsify it in general. The intuition on which it is based is presumably this: The undirected bottom-up strategy has the property that every substring that can be construed as a phrase is identified as such, completely without regard to context. So, for example, the last two words of *radio broadcasts pay* would appear in several lines of the configuration table as a sentence. This does not happen when a top-down procedure is used. In this case, a phrase is sought only when a string of symbols could occur to its right—though possibly not the one actually there—that would complete a sentence containing that phrase. In other words, the search for a phrase of a given category at a given place in the string is undertaken by a top-down parser only if such a phrase could be combined with phrases already found into a legal structure for a sentence beginning with the words of the string under analysis up to the current point. On the other hand, an undirected top-down parser can postulate phrases, and phrases within those phrases, that cannot accomodate even the very next word to be examined. For example, a more ample grammar than (1) would provide for sentences beginning with infinitives, gerunds, and topicalized prepositional phrases, and a top-down parser would attempt all these expansions of the initial $[?]_S$ even though no sentence beginning with phrases of these types could also begin with the word *radio.*

Stronger claims can be made for directed strategies. A situation that favors one or the other kind of undirected parsing method favors all directed methods as against them. Consider, as in transition rule 2, an adjacent pair of items, g_i and g_{i+1} in a configuration, g_i being active with $[?]_\alpha$ as its first open box and g_{i+1} being inactive and of category β. The top-down strategy is to fill the open box in g_i with a string of new open boxes and to continue doing this until the first member of the most recently introduced string is $[?]_\beta$. The number of steps in this process is determined by the grammar and the shaper test together, if the rules involved contain no cycles, and otherwise by the shaper test alone. In any case, once it is set in motion, it takes no further cognizance of the symbols in the string. Thus, if the rules involved do contain cycles or the operation takes place in the early part of a long string, a long train of events may take place without reference to the string being analyzed. Ignoring possible interventions by the shaper, rules of the form $\gamma \rightarrow \beta \ldots$ must be applied at least once during the processing, given that β is the category of the next terminal, for this is precisely what the reachability table (or the rule-selection table) guarantees to parsers employing

directed methods. Let us call these *bottom* rules because they complete the bottom end of a left branching path from the upper symbol, α, to the lower symbol, β.

Under the same initial conditions, a bottom-up procedure starts by applying the bottom rules against the possibility that the same left branching path can be constructed in the reverse order. If successful, this process will finish by applying *top* rules of the form $\alpha \rightarrow \ldots$. But it may not be successful, and therein lies the crucial difference. The left branch will not necessarily be constructed by the bottom-up procedure because the next rule applied after the bottom rule will be the next one in this sequence only if the bottom rule is nonbranching. Otherwise, the second symbol on its right-hand side becomes a new upper symbol whose corresponding lower symbol is the next terminal in the string. In general, the next upward step in the construction of a left branch will be taken only when the most recent symbol has been established at the root of a complete phrase, without open boxes. For every new rule application—every invocation of transition rule 2—made in the bottom-up process, a corresponding application of the same rule, with a view to establishing a phrase at the same place in the overall structure, will be made by the top-down process. The reverse, however, is not true.

Suppose, for example, that the string to be analyzed is *alpha beta gamma delta*, the words belonging to the categories A, B, C, and D respectively. The input to the parser will therefore be $[\text{alpha}]_A[\text{beta}]_B[\text{gamma}]_C[\text{delta}]_D$. Let the grammar be as in (23).

(23)

1. S → P D
2. P → Q C
3. Q → A B

The string has one structure which is readily discovered by all the methods we have discussed with about equal ease. Now consider what happens if the category of the second word is changed so that the string is no longer accepted by the grammar. The first symbol continues to be reachable from the initial symbol of the grammar, S. The top-down strategy proposes rule 2 because the first symbol in the string is also reachable from Q. Then rule 3 is invoked. Its first symbol matches, the second one fails to match, and the process comes to an end, two rules having been considered. The bottom-up strategy calls for rule 3 to be the first one considered. Its first symbol matches the first item in the string, but the second one fails, bringing the whole process to an end after only one rule has been considered. The example could have been constructed so as to make the chain of rules needlessly invoked in the top-down procedure of any desired length.

Contrary to intuitions deriving from better understood undirected methods, it is therefore possible to assert that bottom-up directed methods are superior to all others within this framework. This is falsified only in situations where short sentences must be parsed using grammars with long rules for these are the cases where the functional load born by the shaper is high. Under these circumstances, the shaper may be in a position to curtail the construction of a left branch before the bottom rule

is applied often enough to recommend the top-down method. This argument in favor of directed bottom-up parsing methods does not rest on an analysis of the worst case that can arise because, as far as is known, the methods discussed here cannot be distinguished in that way. The claim is to the effect that such differences in performance as do arise will invariably favor the directed bottom-up technique. *Pace* the use of the shaper, no case can be constructed in which, say, directed top-down parsing fares better.

3. THE CHART

In the last section, our aim was to abstract from a large class of syntactic-processing algorithms those properties of them that ensure that all the processing is done that has to be. We now turn to the problem of eliminating unnecessary repetition of parts of the computation. By far the major part of this repetition comes from a phenomenon that the simple example used was carefully designed to avoid. The phenomenon is this: if a particular initial segment of a long string is analyzable as a phrase of a given type, but in more than one way, then any computation aimed at recognizing phrases in the remainder of the string will be repeated once for each analysis. This is because no mechanism has been provided for recognizing when the phrases required to advance a particular configuration already exist in another.

Consider the sentence *Failing students looked hard.* The first two words can be construed as a noun phrase in two ways meaning approximately *students that fail* and *the failing of students.* Likewise, the second pair of words can be construed as a verb phrase in two ways with the meanings *looked intently* and *looked difficult to do.* These constructions are provided for in the grammar (24). According to this grammar, therefore, the sentence has four structures, each construction of the subject being combinable with each construction of the predicate. But, whereas the parsing strategies so far examined would discover each construction of the subject only once, they would reanalyze the predicate independently for each of these.

(24)

1. S → NP VP
2. NP → A N
3. NP → PRP N
4. VP → V A
5. VP → V AV
6. A → failing, hard, ...
7. PRP → failing, ...
8. N → students, ...
9. V → looked, ...
10. AV → hard, ...

(25) shows the analysis by the undirected bottom-up strategy. The two constructions of the subject are established in lines 7 and 8. A following verb phrase is postulated for each construction

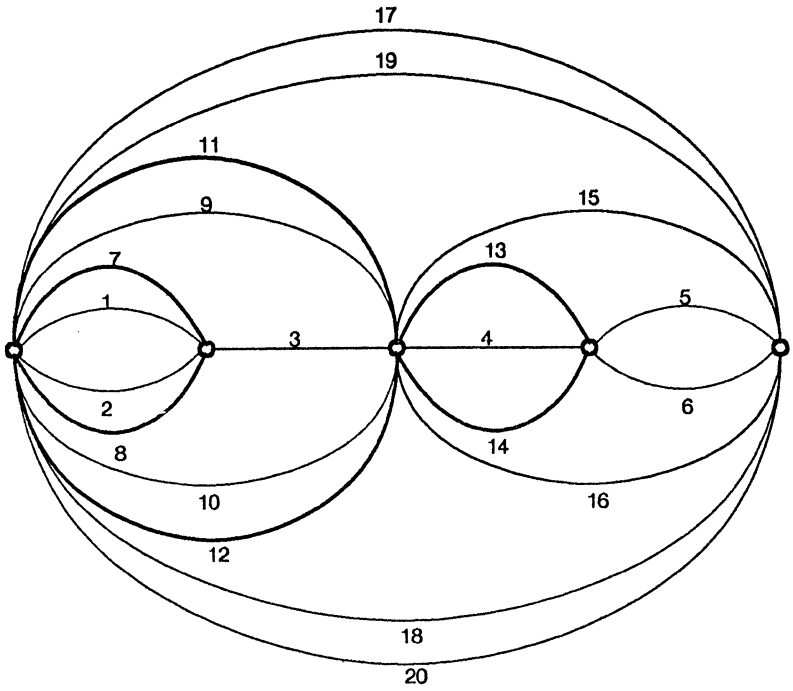
of the subjects in lines 9 and 10 and the verb *looked* is adjoined to each of these configurations in lines 11 and 12. Each of these configurations has two offspring corresponding to the two constructions of the predicate that will eventually be found. From this point on, four lines of development are pursued independently to the final results in lines 29 through 32. Lines 17, 19, 21, and 23 are derived from their parent lines by exactly parallel moves. Those terms that serve to differentiate these lines, but which are not affected by the operation, are therefore brought forward unchanged to lines 22 and 24. These unchanged terms are, in an important sense, *equivalent*, and the new terms introduced are equivalent to one another in the same sense. In this section, I shall first explicate this notion of equivalence and then propose a new data structure, called a *chart*, a kind of well-formed substring table,[3] to replace configuration tables. A chart admits only one member of an equivalence class of terms and, when transitions rules are defined on it corresponding to those defined for configuration

(25)

#	T.R.	Pos.	Par.	R	Configuration
1		1	0		$[\text{failing}]_{A}$
2		1	0		$[\text{failing}]_{PRP}$
3	2	1	1	2	$[[\text{failing}]_{A}[?]_{N}]_{NP}$
4	2	1	2	3	$[[\text{failing}]_{PRP}[?]_{N}]_{NP}$
5	1	2	3		$[[\text{failing}]_{A}[?]_{N}]_{NP}[\text{students}]_{N}$
6	1	2	4		$[[\text{failing}]_{PRP}[?]_{N}]_{NP}[\text{students}]_{N}$
7	3	2	5		$[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}$
8	3	2	6		$[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}$
9	2	2	7	1	$[[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}[?]_{VP}]_{S}$
10	2	2	8	1	$[[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}[?]_{VP}]_{S}$
11	1	3	9		$[[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}[?]_{VP}]_{S}[\text{looked}]_{V}$
12	1	3	10		$[[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}[?]_{VP}]_{S}[\text{looked}]_{V}$
13	2	3	11	4	$[[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}[?]_{VP}]_{S}[[\text{looked}]_{V}[?]_{A}]_{VP}$
14	2	3	11	5	$[[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}[?]_{VP}]_{S}[[\text{looked}]_{V}[?]_{AV}]_{VP}$
15	2	3	12	4	$[[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}[?]_{VP}]_{S}[[\text{looked}]_{V}[?]_{A}]_{VP}$
16	2	3	12	5	$[[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}[?]_{VP}]_{S}[[\text{looked}]_{V}[?]_{AV}]_{VP}$
17	1	4	13		$[[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}[?]_{VP}]_{S}[[\text{looked}]_{V}[?]_{A}]_{VP}[\text{hard}]_{A}$
18	1	4	13		$[[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}[?]_{VP}]_{S}[[\text{looked}]_{V}[?]_{A}]_{VP}[\text{hard}]_{AV}$
19	1	4	14		$[[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}[?]_{VP}]_{S}[[\text{looked}]_{V}[?]_{AV}]_{VP}[\text{hard}]_{A}$
20	1	4	14		$[[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}[?]_{VP}]_{S}[[\text{looked}]_{V}[?]_{AV}]_{VP}[\text{hard}]_{AV}$
21	1	4	15		$[[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}[?]_{VP}]_{S}[[\text{looked}]_{V}[?]_{A}]_{VP}[\text{hard}]_{A}$
22	1	4	15		$[[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}[?]_{VP}]_{S}[[\text{looked}]_{V}[?]_{A}]_{VP}[\text{hard}]_{AV}$
23	1	4	16		$[[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}[?]_{VP}]_{S}[[\text{looked}]_{V}[?]_{AV}]_{VP}[\text{hard}]_{A}$
24	1	4	16		$[[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}[?]_{VP}]_{S}[[\text{looked}]_{V}[?]_{AV}]_{VP}[\text{hard}]_{AV}$
25	3	4	17		$[[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}[?]_{VP}]_{S}[[\text{looked}]_{V}[\text{hard}]_{A}]_{VP}$
26	3	4	20		$[[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}[?]_{VP}]_{S}[[\text{looked}]_{V}[\text{hard}]_{AV}]_{VP}$
27	3	4	21		$[[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}[?]_{VP}]_{S}[[\text{looked}]_{V}[\text{hard}]_{A}]_{VP}$
28	3	4	24		$[[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}[?]_{VP}]_{S}[[\text{looked}]_{V}[\text{hard}]_{AV}]_{VP}$
29	3	4	25		$[[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}[[\text{looked}]_{V}[\text{hard}]_{A}]_{VP}]_{S}$
30	3	4	26		$[[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}[[\text{looked}]_{V}[\text{hard}]_{AV}]_{VP}]_{S}$
31	3	4	27		$[[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}[[\text{looked}]_{V}[\text{hard}]_{A}]_{VP}]_{S}$
32	3	4	28		$[[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}[[\text{looked}]_{V}[\text{hard}]_{AV}]_{VP}]_{S}$

[3]See Kuno (1965).

(26)



(27)

#	Locus	Length	Term
1	0	1	$[\text{failing}]_{A}$
2	0	1	$[\text{failing}]_{PRP}$
3	1	1	$[\text{students}]_{N}$
4	2	1	$[\text{looked}]_{V}$
5	3	1	$[\text{hard}]_{A}$
6	3	1	$[\text{hard}]_{AV}$
7	0	1	$[[\text{failing}]_{A}[?]_{N}]_{NP}$
8	0	1	$[[\text{failing}]_{PRP}[?]_{N}]_{NP}$
9	0	2	$[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}$
10	0	2	$[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}$
11	0	2	$[[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}[?]_{VP}]_{S}$
12	0	2	$[[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}[?]_{VP}]_{S}$
13	2	1	$[[\text{looked}]_{V}[?]_{AV}]_{VP}$
14	2	1	$[[\text{looked}]_{V}[?]_{A}]_{VP}$
15	2	2	$[[\text{looked}]_{V}[\text{hard}]_{AV}]_{VP}$
16	2	2	$[[\text{looked}]_{V}[\text{hard}]_{A}]_{VP}$
17	0	3	$[[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}[[[\text{looked}]_{V}[\text{hard}]_{AV}]_{VP}]_{VP}]_{S}$
18	0	3	$[[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}[[[\text{looked}]_{V}[\text{hard}]_{AV}]_{VP}]_{VP}]_{S}$
19	0	3	$[[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}[[[\text{looked}]_{V}[\text{hard}]_{A}]_{VP}]_{VP}]_{S}$
20	0	3	$[[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}[[[\text{looked}]_{V}[\text{hard}]_{A}]_{VP}]_{VP}]_{S}$

tables in the last section, redundant operations are therefore naturally eliminated.

Undirected Parsing

In what follows, I shall describe two ways of representing in a chart the data that a parsing procedure operates on, one appropriate to undirected and the other to directed parsing. This section is devoted to the first of these. I shall first provide fairly direct translations into the new formalism of the undirected strategies described in the last section. I shall then go on to consider the case in which the grammar is represented as a rule-selection table so that the top-down and bottom-up cases will not have to be distinguished. Of course, the information contained in a rule selection table is largely degenerate in the undirected case. However, we shall see that a nondegenerate table—one designed for a directed procedure—can be combined with chart operations appropriate to undirected processing. The result is a *partially directed* strategy which is not only viable but, in some ways, attractive. If a degenerate rule-selection table—one intended for undirected parsing—is used with chart operations appropriate to directed parsing, the result is simply an inefficient version of undirected parsing with no redeeming features.

I shall refer to the number of terminal symbols represented in a term as the *length* of that term. Open boxes do not count as terminals so that, for example, $[?]_{NP}$ and $[[?]_{Adj}[?]_{N}]_{NP}$ both have lengths of 0; $[[\text{radio}]_{Adj}[?]_{N}]_{NP}$ has length 1, and $[[\text{radio}]_{Adj}[\text{broadcasts}]_{N}]_{NP}$ has length 2. The length of a sequence or a configuration will be the sum of the lengths of the terms in it. The *locus* of a term in a configuration is simply the sum of the lengths of the terms that precede it in the configuration. Intuitively, the locus captures the notion of position in the sentence; two terms have the same locus if they follow the same sequence of words. The first term in a configuration has a locus of 0.

The undirected processing strategies we have considered have the property that the only terms that need be considered when creating a new term with a given locus are exisiting terms with the same or greater loci. This is trivially true of undirected top-down strategies because all constructed terms—that is, all non-terminals—have locus 0. Configurations are of three types: (i) Those consisting of a single inactive term, (ii) Those consisting of a sequence of active terms, and (iii) Those consisting of a sequence of active terms followed by a single inactive term. Now, a crucial fact about a configuration table arising from an undirected procedure is that the configurations making it up are just those that can be constructed from the terms in it, and their loci, subject to the constraints just mentioned. In other words, the set of entries in a configuration table can be generated knowing only the set of different terms that it contains and the loci of each.

Table (25) is a configuration table showing the analysis of *failing students looked hard* by the undirected bottom-up method and grammar (24). A corresponding chart is given at (27). It contains all the terms in (25), and gives the locus and length of each. Now, (27) contains 8 inactive terms

with locus 0, two terminals from the initial string, two noun phrases, one for each interpretation of the subject of the sentence, and four different analyses of the complete sentence. These contribute 8 configurations to (25). 8 different sequences of active terms can be constructed such that the first one has locus 0, four with one member and four with two. 16 sequences can be constructed from active terms followed by a single inactive term. The unambiguous terminals *students* and *looked* are each final members of two of these as are the two verb phrases representing alternative structures of the predicate. Each of the two lexical entries for *hard* is the final member of four sequences. The resulting 32 sequences are precisely the ones that make up the configuration table (25). For undirected strategies, then, a pair of similar terms are equivalent for present purposes if they have the same locus. Table (27) contains only one representative of each equivalence class.

A natural way to represent a chart is as a directed graph. The graph for a sentence of n words has $n+1$ vertices. Terms label edges, and all the terms with a given locus label edges incident from the vertex for that locus. An edge is said to be active or inactive depending on the status of the term that labels it. A term with locus i and length j is incident to the vertex for locus $i+j$. (26) is the graph corresponding to the table (27). The active edges are shown as thicker lines. Arrowheads are not used in the diagrams in this paper because all edges are either oriented from left to right, or loop back to the same vertex.

It remains to restate the transitions rules for undirected parsing so that they apply to a chart instead of a configuration table. The new set of transition rules need have no member corresponding to rule 1 in the old sets. The chart must contain an entry for each lexical entry of each word in the string being analyzed and the locus of each of these is determined simply by the position in the string of the word. Each of these is the sole member of an equivalence class and they can simply be entered in the chart before the analysis proper begins. In (27), they are lines 1 through 6.

For bottom-up parsing, rule 1 of the new set, corresponding to the old rule 2, is:

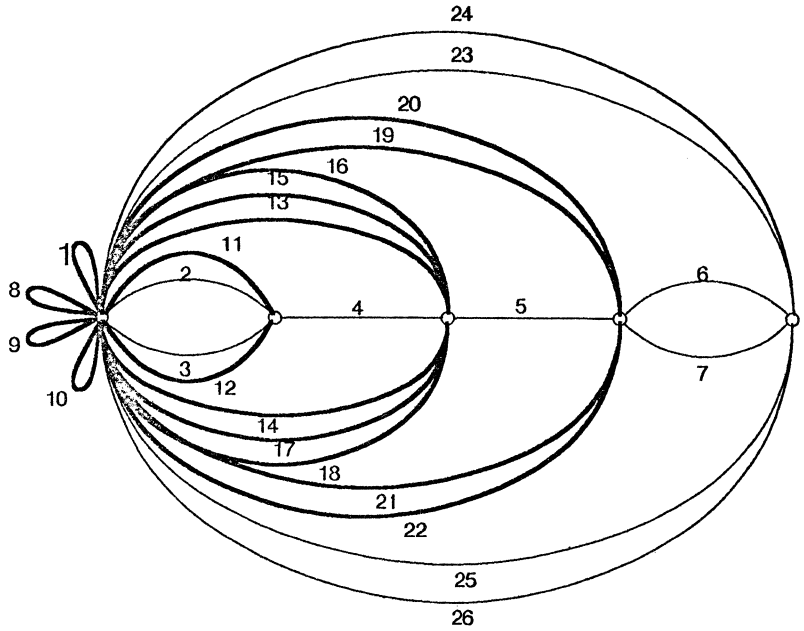
(28) 1. **Find a dominating non-terminal:** Let e_i be an inactive edge of category α incident from a vertex v. For all rules of the form $\beta \rightarrow \gamma_1 \ldots \gamma_n$ in the grammar such that $\gamma_1 = \alpha$, introduce a new edge e_i with the term $[\alpha\ [?]_{\gamma_2} \ldots [?]_{\gamma_n}]_\beta$, incident from and to v, provided there is no such edge in the chart already.

The replacement for Rule 3 must be constrained to apply at only one place in a configuration in the same way. Rule 2 in the new system is thus:

(29) 2. **Incorporate completed subgoal:** Let e_a and e_i be adjacent active and inactive edge. e_a is incident from vertex v and e_i is incident to vertex w. Let $[?]_\alpha$ be the first open box in e_a. If e_i is of category α, create a new edge between v and w whose term is that of e_a with the first open box replaced by the term of e_{i+1}.

Both these rules are applied to all qualifying edges or pairs of edges, including any that arise as a result of applying these rules. All but the initial six entries in (25) will be seen to be generable by these rules and, as we have seen, (25) contains the same information as (27). Unnecessary steps

(30)



(31)

#	Locus	Length	Term
1	0	0	$[?]_S$
2	0	1	$[\text{failing}]_A$
3	0	1	$[\text{failing}]_{PRP}$
4	1	1	$[\text{students}]_N$
5	2	1	$[\text{looked}]_V$
6	3	1	$[\text{hard}]_A$
7	3	1	$[\text{hard}]_A$
8	0	0	$[[?]_{NP}[?]_{VP}]_S$
9	0	0	$[[[?]_A[?]_N]_{NP}[?]_{VP}]_S$
10	0	0	$[[[?]_{PRP}[?]_N]_{NP}[?]_{VP}]_S$
11	0	1	$[[[\text{failing}]_A[?]_N]_{NP}]_S$
12	0	1	$[[[\text{failing}]_{PRP}[?]_N]_{NP}]_S$
13	0	2	$[[[\text{failing}]_A[\text{students}]_N]_{NP}[?]_{VP}]_S$
14	0	2	$[[[\text{failing}]_{PRP}[\text{students}]_N]_{NP}[?]_{VP}]_S$
15	0	2	$[[[\text{failing}]_A[\text{students}]_N]_{NP}[[?]_V[?]_A]_{VP}]_S$
16	0	2	$[[[\text{failing}]_A[\text{students}]_N]_{NP}[[?]_V[?]_{AV}]_{VP}]_S$
17	0	2	$[[[\text{failing}]_{PRP}[\text{students}]_N]_{NP}[[?]_V[?]_A]_{VP}]_S$
18	0	2	$[[[\text{failing}]_{PRP}[\text{students}]_N]_{NP}[[?]_V[?]_{AV}]_{VP}]_S$
19	0	3	$[[[\text{failing}]_A[\text{students}]_N]_{NP}[[\text{looked}]_V[?]_A]_{VP}]_S$
20	0	3	$[[[\text{failing}]_A[\text{students}]_N]_{NP}[[\text{looked}]_V[?]_{AV}]_{VP}]_S$
21	0	3	$[[[\text{failing}]_{PRP}[\text{students}]_N]_{NP}[[\text{looked}]_V[?]_A]_{VP}]_S$
22	0	3	$[[[\text{failing}]_{PRP}[\text{students}]_N]_{NP}[[\text{looked}]_V[?]_{AV}]_{VP}]_S$
23	0	4	$[[[\text{failing}]_A[\text{students}]_N]_{NP}[[\text{looked}]_V[\text{hard}]_A]_{VP}]_S$
24	0	4	$[[[\text{failing}]_A[\text{students}]_N]_{NP}[[\text{looked}]_V[\text{hard}]_{AV}]_{VP}]_S$
25	0	4	$[[[\text{failing}]_{PRP}[\text{students}]_N]_{NP}[[\text{looked}]_V[\text{hard}]_A]_{VP}]_S$
26	0	4	$[[[\text{failing}]_{PRP}[\text{students}]_N]_{NP}[[\text{looked}]_V[\text{hard}]_{AV}]_{VP}]_S$

in the computations have, however, been eliminated because each locus-term pair is computed only once.

Rule 1 for top-down parsing is:

(32) 1. **Find a dominating non-terminal:** Let e_a be an active edge whose first open box is $[?]_\alpha$, incident to a vertex v. For all rules of the form $\alpha \rightarrow \beta_1 \ldots \beta_n$ in the grammar introduce a new edge incident from and to v whose term is the term of e_a with the first open box replaced by $[[?]_{\beta_1} \ldots [?]_{\beta_n}]_\alpha$, provided there is no such edge in the chart already.

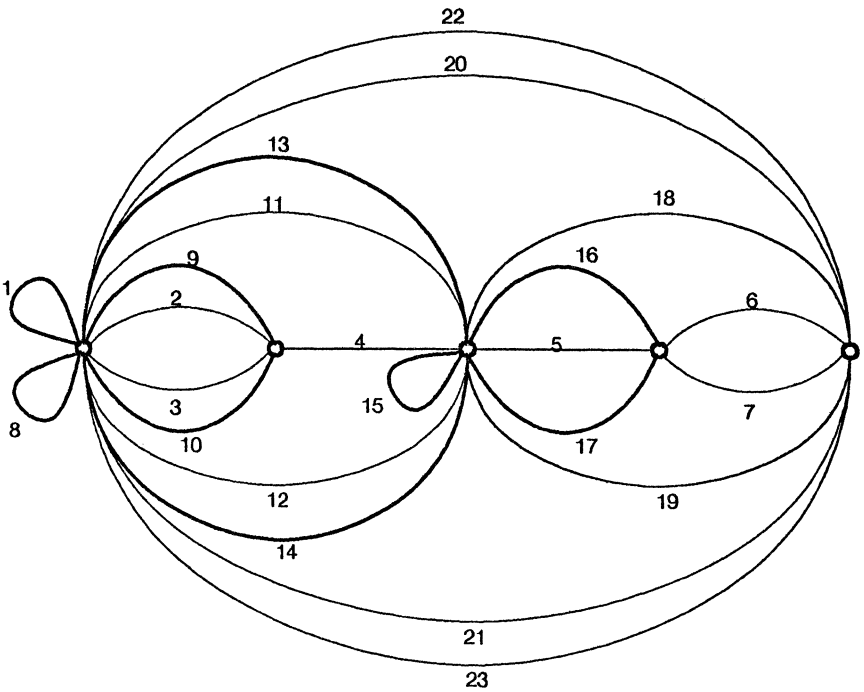
(30) and (31) show charts generated by this method for the same example. I have already remarked that all terms in a top-down configuration table have locus zero. If the transition rules are translated directly, as I have just done, the chart therefore offers little advantage. The reason for this is that, when the transition rule (32) is applied, the new phrase marker is created which fills an open box in a structure with locus 0. In the bottom-up case, the newly introduced phrase marker was not immediately engulfed by a higher structure; in fact it engulfed a lower one and the result could have any locus. The decision to write this transition rule so as to amalgamate the new phrase marker into an existing term is made in the interests of efficiency. As we saw, the procedure that makes use of a rule-selection table inserts a new term into the sequence, leaving it to a later application of the old rule 3 to perform the amalgamation. However, it now appears that the minor logical confusion that comes from confounding the functions of the two transition rules leads to considerable inefficiencies in the chart method.

The procedure that uses a rule-selection table must apply to an adjacent pair of edges, the first active and the second inactive in spite of the fact that the active edge influences the result only in top-down methods and the inactive edge only in bottom-up methods. By excluding pairs both of which are active we implement the policy established previously of applying rule 1 only to one pair of terms in a configuration. For this method, the required rule is given in (33) and (34) and (35) repeat the example with this method.

(33) 1. **Find a dominating non-terminal:** Let e_a and e_i be adjacent active and inactive edges, e_a being incident from v and e_i being incident to w. Let $[?]_\alpha$ be the first open box in e_a and let β be the category of e_i. For all rules $\gamma \rightarrow \delta_1 \ldots \delta_n$ in $S_{\alpha,\beta}$, introduce a new edge between v and w and with the term $[[?]_{\delta_1} \ldots [?]_{\delta_n}]_\gamma$, provided there is no such edge in the chart already.

Examination of (34) is instructive. All applications of transition rule 1 cause a term of length 0 to be introduced and these are recognizable in the graph as active edges that are incident to and from the same vertex. Other edges are either initial lexical edges or arise from applications of rule 2. For this particular example, it is not necessary to state whether the rule-selection table used corresponds to a top-down or a bottom-up strategy because the results are the same in both cases. This, however, is an artefact of the trivial grammar used. The chart does, however, accord equal advantages to both methods both here and in general. Furthermore, nothing turns on whether a

(34)



(35)

#	Locus	Length	Term
1	0	0	$[?]_{S}$
2	0	1	$[\text{failing}]_{A}$
3	0	1	$[\text{failing}]_{PRP}$
4	1	1	$[\text{students}]_{N}$
5	2	1	$[\text{looked}]_{V}$
6	3	1	$[\text{hard}]_{A}$
7	3	1	$[\text{hard}]_{AV}$
8	0	0	$[?]_{NP}$
9	0	1	$[[\text{failing}]_{A}[?]_{N}]_{NP}$
10	0	1	$[[\text{failing}]_{PRP}[?]_{N}]_{NP}$
11	0	2	$[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}$
12	0	2	$[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}$
13	0	2	$[[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}[?]_{VP}]_{S}$
14	0	2	$[[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}[?]_{VP}]_{S}$
15	2	1	$[?]_{VP}$
16	2	1	$[[\text{looked}]_{V}[?]_{AV}]_{VP}$
17	2	1	$[[\text{looked}]_{V}[?]_{A}]_{VP}$
18	2	2	$[[\text{looked}]_{V}[\text{hard}]_{AV}]_{VP}$
19	2	2	$[[\text{looked}]_{V}[\text{hard}]_{A}]_{VP}$
20	0	3	$[[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}[[[\text{looked}]_{V}[\text{hard}]_{AV}]_{VP}]_{VP}]_{S}$
21	0	3	$[[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}[[[\text{looked}]_{V}[\text{hard}]_{AV}]_{VP}]_{VP}]_{S}$
22	0	3	$[[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}[[[\text{looked}]_{V}[\text{hard}]_{A}]_{VP}]_{VP}]_{S}$
23	0	3	$[[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}[[[\text{looked}]_{V}[\text{hard}]_{A}]_{VP}]_{VP}]_{S}$

rule-selection table is in fact used; the crucial requirement is that the functions of the two transition rules should remain dissociated.

There is one respect in which the chart procedures using a rule-selection table fails to reflect the top-down parsing strategy accurately. This arises when there are sets of left-recursive rules. Careful examination of the transition rules shows that the phrases will not be built in a strictly top-to-bottom order in this case but that the phrases constructed in the bottom cycle of such a set of rules will be produced first. The topmost member of the set of phrases will then serve as a terminal and the cycle will repeat. This happens because the zero-length edge introduced by transition rule 2 for each rule in the set is the same each time; it incorporates nothing from the edges on the basis of which it was introduced. There will be only one such edge for each rule in the left-recursive set regardless of how many cycles of the set take place at a given point in the analysis. The only move open, once such an edge has been established is therefore to pursue the downward path. One consequence of this is that the shaper test is no longer required because the deviation from the top-down strategy occurs in just such a way as to break the vicious cycle.

Directed Parsing

Representing the state of a parsing device by means of a chart can bring the same advantages to directed as to undirected parsing. However, the chart must be used differently because the equivalence classes of terms that are of interest now are not simply those that have the same locus. The point is best illustrated by a string with more than one left branching structure. Consider the sentence *Can openers work* which, punctuation aside, can be read as an assertion about can openers or a question about openers. The first word presumably has two lexical entries, one as a modal verb, and the other as a noun. In its capacity as a modal, we may assume it can be an auxiliary and, as a noun, it can become a noun phrase. I shall refrain from attempting to write explicit context-free rules for the relevant parts of English grammar. Relations between concision and verisimilitude have already been sufficiently strained in this paper.

Using a reachability, or rule-selection, table, the parser posits an initial noun phrase on the basis of the first noun and an auxiliary on the basis of the modal. Four questions now arise: (i). Can the modal begin an auxiliary? (ii) Can the noun begin a noun phrase? (iii) Can the modal begin a noun phrase? and (iv) Can the noun begin an auxiliary? But clearly only the first two of these are proper questions to be considered as part of a directed strategy. Now, it is true that a number of questions that an undirected parser would have to consider are avoided. An undirected top-down parser would, for example establish hypotheses—configurations—that could only be born out if the first word was a determiner, an infinitive or, indeed, any category that can begin a sentence. Each of these would be eventually ruled out when neither lexical entry for the first word matched them. Similary, an undirected bottom-up parser, observing that *can* can be a main verb, would go on to discover that *can openers* is a verb phrase.

The reason that two irrelevant questions arise is that the chart puts all terms with a given locus on the same footing. In the purportedly directed procedure we are now investigating, grammar rules will therefore be applied to all terms with a given locus provided the reachability table sanctions their application to any term with that locus. As more phrases are formed, more unnecessary work is done. Later in the analysis, it will emerge that the first two words, *can openers* constitute a noun phrase and, at that point, the question of whether this could be initial in an auxiliary will be considered. In short, a procedure that uses a reachability, or rule-selection, table together with the chart-management regime just decribed, is only partially directed.

I shall now show that the chart representation can easily be made to support fully directed context-free parsers. The crucial observation has just been made, namely that terms with the same locus do not constitute the interesting equivalence classes for directed processing. Each class must now contain terms at a given locus and with a given category. But, the full generality of the chart representation will be preserved only if there continues to be a vertex corresponding to each locus. This is the vertex to which terms whose successors have that locus will be incident. For ease of reference, I shall call these *locus vertices.* Those from which edges representing terms of a certain category are incident will be called *term vertices.* Two kinds of edge will also be required. The *term edges*, familiar from the last section, will always be incident from *term vertices.* Inactive term edges will be incident to locus vertices and active term edges to term vertices. The term vertices for a given locus will be accessible from the corresponding locus vertex by following a *category edge* whose label, instead of being a term, will simply be the category characteristic of the term vertex in question.

Each lexical entry for a string to be parsed is represented in the new chart by a pair consisting of a category edge followed by a term edge. The category edges for the i-th word are incident from locus vertex i. If α is a category of the i-th word, a category edge will be constructed from the locus edge i to the term vertex $i : \alpha$, and from there a term edge, labeled with the lexical entry, will be constructed to the locus vertex $i + 1$. Suppose now that an arbitrary inactive term is to be entered into the chart at locus L. Let its length be ℓ, and its category α. The edge that it labels will be incident from the term vertex $L : \alpha$, and to the locus vertex $L + \ell$. If vertex $L : \alpha$ did not previously exist, it would be necessary, at the same time, to create a category edge giving access to that vertex from the locus vertex L.

The parsing procedure starts when an active edge labeled with the open box $[?]_S$ is entered into the chart, incident from and to the locus vertex 0. From this point on, the process is governed by (36).

(36) 1. **Find a dominating non-terminal:** Let e_a be an active edge incident from vertex v to the vertex w, and e_i an inactive edge of category ϵ incident from w. Let L be the locus of w or, more strictly, of the edges incident from w. (If e_i is a category edge, ϵ is its label and L is the name of the vertex w; if e_i is a term edge, its label is of the form $[\ldots]_\epsilon$ and the

(37)

#	Left	Right	Term
1	0	0:A	A
2	0:A	1	$[\text{failing}]_{A}$
3	0	0:PRP	PRP
4	0:PRP	1	$[\text{failing}]_{PRP}$
5	1	1:N	N
6	1:N	2	$[\text{students}]_{N}$
7	2	2:V	V
8	2:V	3	$[\text{looked}]_{V}$
9	3	3:A	A
10	3:A	4	$[\text{hard}]_{A}$
11	3	3:AV	AV
12	3:AV	4	$[\text{hard}]_{AV}$
13	0	0:S	S
14	0:S	0:A	$[?]_{S}$
15	0:S	0:PRP	$[?]_{S}$
16	0	0:NP	NP
17	0:S	0:NP	$[[?]_{NP}[?]_{VP}]_{S}$
18	0:NP	0:A	$[[?]_{A}[?]_{N}]_{NP}$
19	0:NP	0:PRP	$[[?]_{PRP}[?]_{N}]_{NP}$
20	0:NP	1	$[[\text{failing}]_{A}[?]_{N}]_{NP}$
21	0:NP	1	$[[\text{failing}]_{PRP}[?]_{N}]_{NP}$
22	0:NP	1:N	$[[\text{failing}]_{A}[?]_{N}]_{NP}$
23	0:NP	1:N	$[[\text{failing}]_{PRP}[?]_{N}]_{NP}$
24	0:NP	2	$[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}$
25	0:NP	2	$[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}$
26	0:S	2	$[[[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}]_{NP}[?]_{VP}]_{S}$
27	0:S	2	$[[[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}]_{NP}[?]_{VP}]_{S}$
28	0:S	2:VP	$[[[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}]_{NP}[?]_{VP}]_{S}$
29	2:VP	2:V	$[[?]_{V}[?]_{A}]_{VP}$
30	0:S	2:VP	$[[[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}]_{NP}[?]_{VP}]_{S}$
31	2:VP	2:V	$[[?]_{V}[?]_{AV}]_{VP}$
32	2:VP	3	$[[\text{looked}]_{V}[?]_{A}]_{VP}$
33	2:VP	3	$[[\text{looked}]_{V}[?]_{AV}]_{VP}$
34	2:VP	3:A	$[[\text{looked}]_{V}[?]_{A}]_{VP}$
35	2:VP	3:AV	$[[\text{looked}]_{V}[?]_{AV}]_{VP}$
36	2:VP	4	$[[\text{looked}]_{V}[\text{hard}]_{A}]_{VP}$
37	2:VP	4	$[[\text{looked}]_{V}[\text{hard}]_{AV}]_{VP}$
38	0:S	4	$[[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}[[\text{looked}]_{V}[\text{hard}]_{A}]_{VP}]_{S}$
39	0:S	4	$[[[\text{failing}]_{A}[\text{students}]_{N}]_{NP}[[\text{looked}]_{V}[\text{hard}]_{AV}]_{VP}]_{S}$
40	0:S	4	$[[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}[[\text{looked}]_{V}[\text{hard}]_{A}]_{VP}]_{S}$
41	0:S	4	$[[[\text{failing}]_{PRP}[\text{students}]_{N}]_{NP}[[\text{looked}]_{V}[\text{hard}]_{AV}]_{VP}]_{S}$

label of w is $L:\epsilon$). Let the term of e_a be of category α and its first open box be $[?]_{\beta}$. For each rule $\gamma \rightarrow \delta_1 \ldots \delta_n \in \mathcal{R}_{\beta,\epsilon}$, create whatever is not already present in the chart of the following structure: (i) the term vertex $L:\gamma$, (ii) a term edge with the same label as e_a from v to $L:\gamma$, (iii) a term edge labeled $[\mathcal{F}(\delta_1)\ldots\mathcal{F}(\delta_n)]_{\gamma}$ from $L:\gamma$ to $L:\epsilon$, and (iv) a category edge labeled γ from w to $L:\gamma$.

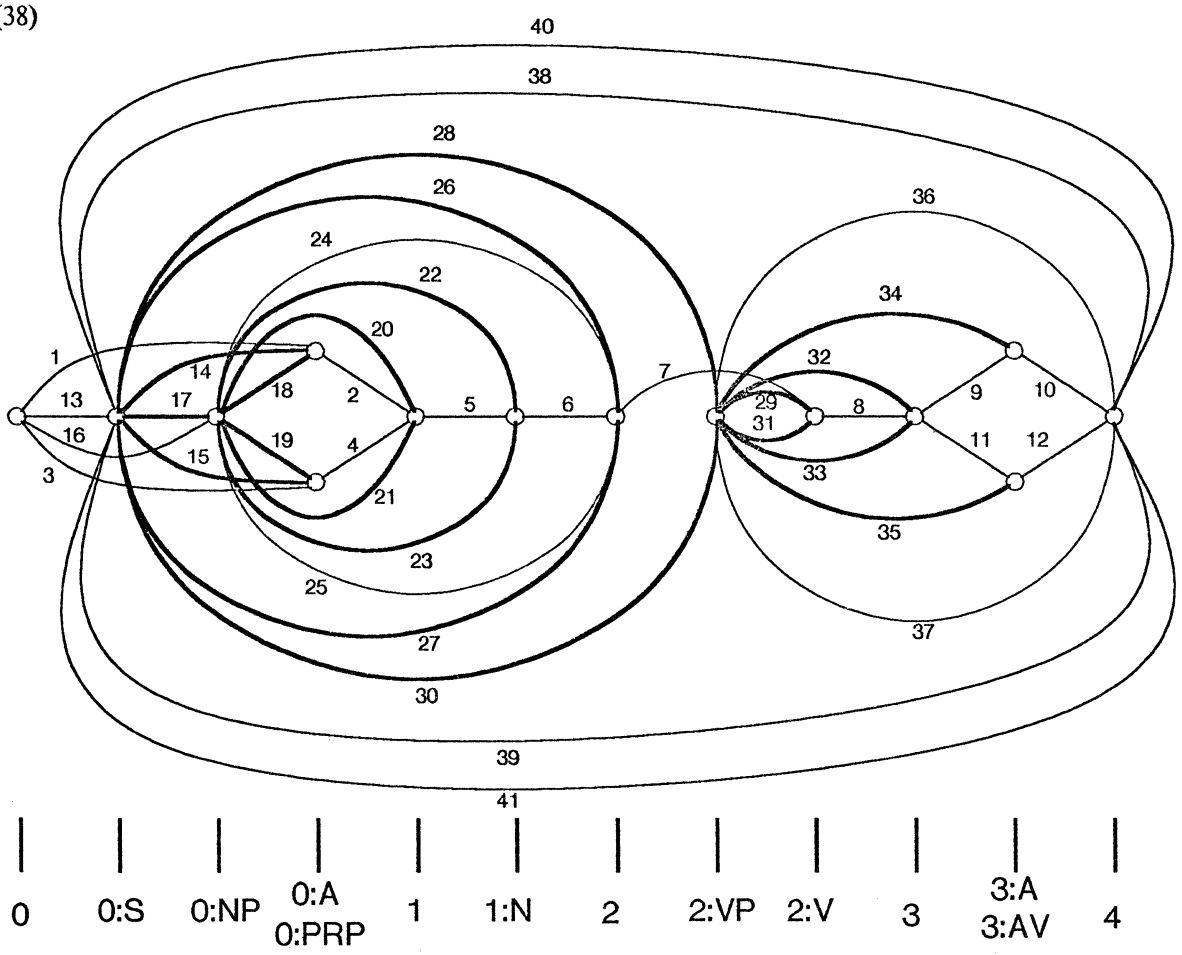
2. **Extend Active Edge:** Let e_a be an active edge incident from vertex v to the locus vertex

L, and e_i a category edge labeled α incident from L to $L:\alpha$. If $[?]_\alpha$ is the first open box in e_a, create a new edge whose term is the same as that of e_a from v to $L:\alpha$.

3. **Incorporate completed subgoal:** Let e_a be an active edge incident from vertex v to the term vertex $L:\alpha$, and e_i a term edge of category α incident from $L:\alpha$ to x. If $[?]_\alpha$ is the first open box in e_a, create a new edge whose term is that of e_a with the first open box replaced by the label of e_i incident from v and to x.

(38) and (37) show the results of analyzing *failing students looked hard* using the grammar at (24) and the directed bottom-up method. Each transition rule operates on a pair consisting of an active edge and an immeadiately following inactive edge and, wherever a rule is applicable to such a pair, it must be applied before the process is complete. A measure of the amount of work done by the procedure at each vertex is therefore given by the product of the number of active edges incident to it and the number of inactive vertices incident from it. The sum of these measures for this example is 24. The comparable result for (34) and (35), the undirected counterpart of this procedure is 28. The small difference in the results is due mainly to the triviality of the example. Directed methods,

(38)



and particularly directed bottom-up methods, come into their own in the face of structures with long left branches.

I believe that the superiority of directed bottom-up parsing methods have not been generally appreciated. However, I also believe that they may not always be appropriate or that the advantages that they offer over simpler undirected methods may not be enough to recommend them in all circumstances. One argument for simpler techniques is based on the presumed nature of natural languages. The number of categories that linguists have thought it right to use in their grammars has usually been very small[4] and the left branches that are constructed without repeating symbols are very short so that, empirically, the advantages of the directed schemes are minimal.

The value of the information in the reachability or rule-selection tables is further debased if rules are stored as transition networks. This method of representing rules has the advantage of conflating subsequences of symbols that occur in more than one rule. In particular, if the right-hand sides of a pair of rules have a common initial substring, then any work that a parser does comparing these symbols against a string contributes to the matching of both rules. However, such a pair of rules may well have different symbols on the left-hand side so that there is no reason to suppose that they would be called upon under similar circumstances by a directed parser.

Finally, as I pointed out at the outset, the current interest in context-free processing techniques is due in part to the fact that they can often be applied to grammatical formalisms that are not strictly context free, for example, various formalisms involving complex symbols. Under these circumstances, the appropriateness of directed processing methods is governed in large measure by whether it is easy, or indeed possible, to construct reachability tables or rule-selection tables within these formalisms.

4. The Agenda

An algorithm schema becomes an algorithm when all the choices that must be made in executing it are determined. The indeterminacy arises because the transition rules we have considered call for some action to be taken whenever an edge of a certain kind is created or, more importantly, whenever a pair consisiting of an active edge followed by an inactive edge comes into being. The actions that take place under these circumstances can easily give rise to new instances of the same kinds of circumstance. In other words, each step in the process can produce the conditions for more than one subsequent step. Clearly, entering one new edge into a chart can increase by an arbitrary amount the number of active-inactive pairs that there are. It is therefore a fundamental feature of parsing that the material to be processed is amassed, during certain phases of the enterprise, faster than it can be treated.

[4]Gazdar (1979a and 1979b) are notable exceptions to this.

A classical response to this problem is through recursion. Suppose there is a procedure whose job is to add new edges to the chart and then to rehearse the new situations thus created to which transition rules must be applied. It applies the rules, calling itself recursively to add new edges as necessary. In general, some number of edges or pairs of edges will be waiting for treatment by transition rules. The queue is maintained by the recursion mechanism. What results is a so-called *depth first* algorithm in which the most recently proposed task is always the next one to be carried out. In parsing, the result is that one, essentially arbitrary, hypothesis about the grammatical structure of the string at hand is pursued relentlessly until it either succeeds or can be pursued no further.

Another technique which preserves the conceptual clarity and operational simplicity of the chart involves a list of *tasks* called an *agenda* and a policy for managing it. A task is simply an edge, or pair of edges—whatever the transition rules require. Whenever new edges are added to the chart, any new tasks that can be created as a result, are added to the agenda. If they are always added to the front of the agenda and are also removed from the front of the agenda for execution, the same depth-first strategy just described results. If they are added to one end of the agenda, but removed from the other, so that the agenda serves as a queue in the usual sense, the result is a *breadth-first* strategy in which, roughly speaking, equal time is given to each currently open hypothesis. Work on a given hypothesis is resumed only when those corresponding to all others have been cycled through.

The agenda, like the other data structures I have proposed, has the appeal of leaving as many options as possible open. Observe that, if the aim of the parsing exercise is to discover all the structures of a potentially ambiguous string of symbols, tasks can be removed from the agenda in any order whatsoever, in as nearly a random way as can be contrived, without influencing the results eventually arrived at or the time taken to reach them. All the tasks must be done and the cost of executing a given task is not affected by the others on the agenda. If, on the other hand, some significance is attached to the order in which solutions are arrived at, then the policies governing the management of the chart become crucially important. Suppose one holds, for example, that the grammar that represents a person's linguistic competence assigns some number of interpretations to a string but that only certain of these will be recognized under conditions of actual performance. Presumably such facts would be explained by positing a specific parsing algorithm or the agenda-management policy that gives rise to that algorithm.

Before pursuing this line too far, we shall do well to assess what the true value of an algorithm as opposed to an algorithm schema actually is. The important fact is that computers, as they exist today, are algorithmic machines *par excellence.* They require algorithms to direct their operations and they are, in principle, able to execute all algorithms indifferently. But machines can be envisaged, and some are being built, which force some reassessment of the notion of an algorithm in the direction suggested in this paper. The main reason for requiring that the order of events be exactly specified is that the computer can do only one thing at once and everything turns on knowing what that shall

be. If a machine were able to do more than one thing at once and, if the outstanding tasks were independent of one another, then a number of them could be under way at any moment and no importance would attach to whether one was finished before or after another. Such a machine might be thought of as having a number of work stations at each of which tasks of the same kind could be done. But some could be faster and some slower; some work stations could be removed from service and others introduced without influencing the outcome. The interest of all this for present purposes has nothing to do with possible future directions in computer science but simply with the very real possibility that algorithms, narrowly construed, are not necessarily what is required as mental models. Indeed, the proposition that the brain is patterned after the computer that we know today is, *prima facie*, implausible.

On the other hand, a standard computer can be set to predict something of the behavior of mental models which countenance the simultaneous execution of several tasks. For example, suppose it was proposed that syntactic tasks all took unit time and that some fixed number, say k, of them could be in process at any time. A computer model of this behavior would maintain a main agenda and a secondary agenda. Tasks would be taken from the secondary agenda in any convenient order for execution and, as new tasks were produced, they would be appended to the main agenda. When the secondary agenda was exhausted, k tasks would be chosen from the main agenda according to a policy laid down as part of the model and transferred to the secondary agenda. If the main agenda only contained k or less tasks, they would all be transferred. The model would predict that the time taken to analyze a string would be proportional, not to the number of tasks executed, but to the number of times the secondary agenda was replenished from the main agenda. The explanatory power of a model which specifies more is surely greater than one that specifies less, but this is not to say that a model is to be preferred that makes von Neuman seem to have invented the brain.

Psychological experiments aimed at obtaining data about syntactic processing strategies are carefully designed to control for any possible intrusion by other mental processes, particularly semantic effects. It seems reasonable and, indeed, it is often observed, that the syntactic reading that first suggests itself to a subject is conditioned by previous context, the meanings of the words involved, and such like. A psychological model must eventually explain this behavior as well as what happens when these effects are, to the extent possible, avoided. A model that took the form of an algorithm, in the narrow sense, would be poorly placed to do this. In fact, the advocate of such a model would be forced into the position that the strategies used in controlled psychological experiments were unrelated to those used in understanding sentences in context. A model based on the agenda can associate priorities with tasks in more or less complex ways and can thus ascribe the variation observed in experimental results to a variety of sources. The simpler experiments reveal what the priorities are when there is little or no semantic contribution but, when the time comes to give an account of that contribution, the work will center on how the priorities are computed and not on designing a completely new algorithm.

An objection that might be raised against a psychological model based on charts and agendas is that it assumes that the result of every intermediate result involved in the overall computation is remembered and equally accessible at any later time. Indeed, the conceptual clarity of this way of doing things comes largely from the fact that there are almost no state variables in the abstract machine. The quantities that such variables would contain in other processors here become permanent parts of a data structure. This argument would be persuasive against a proposal that charts and agendas as described here be accorded the status of a psychological model. But that would be an absurd proposal. High on the long list of additions that would have to be made to it before it could fill that role would be a model of short term memory that would have a notion of time and of the intervals of time during which particular parts of each data structure were accessible. But the notion of time would not be taken over uncritically from the instruction counter of the computer. In short, what I have presented here is what I claimed it would be, namely a framework within which parsers, conceptual and actual, can be designed and constructed. The choices that it invites the designer to make are choices that are motivated by inherent properties of the problem of syntactic processing and not by more or less irrelevant considerations from automata theory and electrical engineering.

REFERENCES

1. Bresnan, J. (1978) "A Realistic Transformational Grammar" in M. Halle, J. Bresnan and G. A. Miller (eds.) *Linguistic Theory and Psychological Reality.* MIT Press.
2. Frazier, L. (1978) "On comprehending sentences: syntactic parsing strategies". Unpublished doctoral dissertation, University of Connecticut.
3. Frazier, L. and J. D. Fodor (1978), "The sausage machine: a new two-stage parsing model", *Cognition 6*, 291-325.
4. Gazdar, G. (1979a), "English as a context-free language", University of Sussex, Mimeo.
5. Gazdar, G. (1979b), "Constituent structures", University of Sussex, Mimeo.
6. Griffiths, T. V. and S. R. Petrick (1965), "On the relative efficiencies of context-free grammar recognizers", *CACM 8, 5*, 289-300.
7. Harman, G. H. (1963), "Generative grammars without transformation rules: a defense of phrase structure", *Language* 39, 4, pp. 597-616.
8. Kaplan, R. M. (1972), "Augmented transition networks as psychological models of sentence comprehension", *Artificial Intelligence, 3*, 77-100.
9. Kaplan, R. M. (1978) "Computational resources and linguistic theory", *TINLAP 2.*
10. Kay, M. (1977) "Moprphological and syntactic analysis" in A. Zampolli (ed.), *Syntactic Structures Processing.* North Holland.
11. Kimball, J. (1973) "Seven principles of surface structure parsing in natural language". *Cognition 2*, 15-47.

12. Kuno, S. and A. Oetinger (1962), "A multiple path syntactic analyzer". *Information Processing 62.* North Holland.
13. Kuno, S. (1965) "The predictive analyzer and a path elimination technique. *CACM, 8,* 453-462
14. Marcus, M. P. (1978) "A theory of syntactic recognition for natural language", Unpublished doctoral dissertation, MIT.
15. Woods, W. A. and R. M. Kaplan (1971), "The lunar sciences natural language information system", Bolt, Beraneck and Newman.
16. Wanner, E., R. Kaplan and S. Shiner (1975) 'Garden paths in relative clauses". Unpublished Paper. Harvard University.
17. Wanner, E. and M. Maratsos (1978) "An ATN approach to comprehension". in M. Halle, J. Bresnan and G. A. Miller (eds.) *Linguistic Theory and Psychological Reality.* MIT Press.

XEROX

XEROX

Algorithm Schemata and Data Structures in Syntactic Processing

by Martin Kay

Xerox Corporation
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, California 94304

XEROX® is a trademark of XEROX CORPORATION Printed in U.S.A.

CSL-80-12